Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2009

Bruno Miguel Marques Ribeiro

Detecção de objectos em robótica usando informação morfológica

Object detection in robotics using morphological information

Bruno Miguel Marques Ribeiro

Detecção de objectos em robótica usando informação morfológica

Object detection in robotics using morphological information

dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor António José Ribeiro Neves, Professor Auxiliar Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor Armando José Formoso de Pinho, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

À Andreia Filipa...

o júri

presidente

Doutor Tomás António Mendes Oliveira e Silva
Professor Associado da Universidade de Aveiro

Doutor António Fernando Macedo Ribeiro
Professor Associado da Universidade do Minho

Doutor Armando José Formoso de Pinho
Professor Associado da Universidade de Aveiro

Doutor António José Ribeiro Neves
Professor Auxiliar Convidado da Universidade de Aveiro

agradecimentos

Aos meus pais que sempre me apoiaram, em especial à melhor mãe do mundo. Ao meu mano Marco que, sendo para mim um exemplo, acreditou sempre no meu potencial e deu forças para que eu chegasse mais longe. À minha namorada Andreia Filipa pelo companheirismo e por ter estado presente quando mais precisei. Ao meu orientador, Doutor António Neves, por ter sido um verdadeiro orientador, por ter acreditado no meu trabalho e por toda a compreensão. Ao meu co-orientador, Doutor Armando Pinho, pela fantástica ajuda na escrita desta obra. A todos os membros da equipa CAMBADA, verdadeiros campeões, com especial agradecimento aos visionários Tozé e Daniel, pela ajuda, confiança e motivação demonstradas.

E a todos aqueles que, de alguma forma, contribuíram para a realização deste trabalho.

palavras-chave

Visão robótica, câmaras omnidireccionais, transformada de Hough, detecção de contornos, detecção morfológica da bola, análise de imagem, biblioteca OpenCV.

resumo

Uma das componentes mais importantes em sistemas de processamento de imagem é a detecção de objectos de interesse. Contudo, a detecção de objectos é um desafio. Dada uma imagem arbitrária e assumindo que se está interessado em localizar um determinado objecto, o grande objectivo da detecção de objectos passa por determinar se existe ou não qualquer objecto de interesse. Esta tese encontra-se inserida no domínio do RoboCup e foca o desenvolvimento de algoritmos para a detecção de bolas oficiais da FIFA, um objecto importante no futebol robótico. Para atingir o objectivo principal, foram desenvolvidos três algoritmos para detectar bolas de futebol com cores arbitrárias, usando informação morfológica obtida através do detector de cortornos $Canny$ e da tranformada de $Hough$. Em primeiro lugar, foi desenvolvida uma abordagem onde se implementou um algoritmo específico usando a transformada de $Hough$ circular. Em segundo lugar, foi implementado um algoritmo que utiliza uma função da biblioteca OpenCV dedicada à procura de círculos em imagens. Finalmente, os dois primeiros algoritmos foram agrupados para criar uma nova abordagem, na qual ambos os algoritmos são usados. São apresentados resultados experimentais que mostram que os algoritmos desenvolvidos são precisos, sendo capazes de realizar a detecção da bola de forma confiável em situações de tempo-real.

keywords Robotic vision, omnidirectional cameras, Hough transform, edge detection, morphological ball detection, image analysis, OpenCV library.

abstract One of the most important steps in image processing systems is the detection of objects of interest. However, object detection is a challenging task. Given an arbitrary image and assuming that we are interested in locating a particular object, the goal of object detection is to determine whether or not there is any object of interest. This thesis is inserted in the RoboCup domain and is focused on the development of algorithms for the detection of arbitrary FIFA balls, an important object for soccer robots. To achieve the main objective, we developed three algorithms to detect arbitrary soccer balls using morphological information given by the Canny edge detector and the Hough Transform. First, it was developed an approach where we implemented a specific algorithm using the circular Hough Transform, applied after the segmentation of the acquired image. Secondly, it was implemented an algorithm that uses a function of the OpenCV library dedicated to the search of circles in images. Finally, the two first algorithms were joined to create a new approach in which both of the algorithms are used. Experimental results are presented, showing that the developed algorithms are accurate, being capable of reliable ball detection in real-time situations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding the environment is a fundamental problem in the design of autonomous mobile robots. A basic part of perception is to learn, detect and recognize objects, which must be done respecting the limited resources of a mobile robot and the limited choice of available sensors. The performance of a mobile robot crucially depends on the accuracy, duration and reliability of its perceptions and the involved interpretation process. Several autonomous robots use a computer vision system to understand the environment around it.

A computer vision system processes images acquired by a digital camera. This process resembles the human vision system, where the brain processes images derived from the eyes. The human vision is a sophisticated system that senses and acts on *visual stimuli*. Intuitively, computer and human vision appear to have the same function. The purpose of both systems is to interpret spatial data, indexed by more than one dimension. Even though computer and human vision are functionally similar, we cannot expect a computer vision system to replicate exactly the function of the human eye. This is partly because we do not fully understand how the eye works and, therefore, we are still unable to replicate exactly the human vision system.

A basic computer vision system requires a camera, a communication channel and a computer. This system, when applied to a robot, becomes the robot vision system. The human perception has the capability to acquire, integrate and interpret all the abundant visual information. It is challenging to impart such capabilities to a machine in order to interpret the visual information embedded in still images, graphics and video in our sensory world.

The first step for designing a digital image analysis system is performing image acquisition

using sensors such as digital cameras. A two-dimensional image that is recorded by these sensors is the mapping of the three-dimensional visual world. The captured two dimensional signals are sampled and quantized to create digital images. This image can be considered as a grid of numbers. Any given number within that grid has a rather large noise component and so, by itself, gives us little information. The task then becomes to turn this noisy grid of numbers into the object of interest.

Another important step in any image understanding system is locating significant objects. However, object detection is a challenging task, because of the variability in scale, location, orientation and pose of the instances of the object in which we are interested. Moreover, occlusions and light conditions also change the overall appearance of objects in images. Given an arbitrary image and assuming to be interested in locating a particular object, the goal of object detection is to determine whether or not there is any object of interest and, if present, return the image location and extend of each instance of the object.

Most image processing and computer vision techniques are implemented in computer software. Nowadays, it is common the use of different processing units like Digital Signal Processors (DSPs), Field-programmable Gate Arrays (FPGAs) or even Graphics Processing Units (GPUs). Regarding the software implementations, C and C++ are yet the most popular languages for vision system implementation. C programming is chosen in several applications because of its strengths in integrating high and low level functions, and the availability of good compilers. As the systems become more complex, C++ becomes more attractive when encapsulation and polymorphism may be exploited.

Nowadays, there are many vision systems in routine industrial use: digital cameras inspecting mechanical parts to check size or quality, driver assistant systems using vision in the car industry, several academic projects around the world, namely the RoboCup organization or the DARPA [1] challenge, as well as several other projects on autonomous robots, namely Honda's ASIMO [2], SONY AIBO [3], Aldebaran Nao [4] (currently the humanoid robot used in the Standard Platform League in the RoboCup competitions), Robotis Bioloid [5] and the Lego Mindstorms [6].

[1] `http://www.darpagrandchallenge.com`
[2] `http://asimo.honda.com`
[3] `http://support.sony-europe.com/aibo`
[4] `http://www.aldebaran-robotics.com/eng/Nao.php`
[5] `http://www.robotis.com/zbxe/bioloid_en`
[6] `http://mindstorms.lego.com`

Moreover, forensic studies and biometrics (ways to recognize people) using computer vision include automatic face recognition and recognizing people by the 'texture' of their irises. These studies are paralleled by biologists and psychologists, who continue to study how the human vision system works and how we see and recognize objects (and people).

In this work, we developed an efficient vision system for an autonomous robot, designed to play football in the RoboCup competitions. In particular, we focused on the development of algorithms for the detection of arbitrary FIFA balls, an important object for soccer robots. However, the algorithms presented in this thesis can be adapted for the detection of other objects of interest.

Using the algorithms presented in this thesis, the CAMBADA robotic soccer team achieved the first place in the mandatory challenge of the Middle Size League in the RoboCup'2009, held in Graz, Austria. In this challenge, the robots have to play with an arbitrary FIFA ball and the main objective is to encourage teams to improve their vision systems.

1.1 The Robocup Federation

The RoboCup Federation [7] is an international organization, registered in Switzerland, to organize international effort to promote science and technology using soccer games by robots and software agents. The RoboCup Federation organizes RoboCup, now called "RoboCup World Championship and Conference".

RoboCup is an international research and education initiative. It is an attempt to foster artificial intelligence and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined, as well as being used for integrated project-oriented education.

Currently, RoboCup has the following domains:

- **RoboCupSoccer** – The soccer game is used as a standard problem, aiming at innovations to be applied for socially significant problems and industries.
- **RoboCupRescue** – One major application of RoboCup technologies is search and rescue in large scale disaster situations. RoboCup initiated the RoboCupRescue project to

[7] `http://www.robocup.org`

specifically promote research in socially significant issues. RoboCupRescue includes real robot and simulation leagues.

- **RoboCup@Home** – RoboCup@Home focuses on real-world applications and human-machine interaction with autonomous robots. The aim is to foster the development of useful robotic applications that can assist humans in everyday life.
- **RoboCupJunior** – RoboCupJunior is a robotics event for primary and secondary school students that provides consistent challenges from year to year and emphasizes sharing ideas in a friendly learning environment.

RoboCup is a task for a team of multiple fast-moving robots under a dynamic environment. In order for a robot team to actually perform a soccer game, various technologies must be incorporated. Such technologies include: real-time sensor fusion, reactive behavior, strategy acquisition, learning, real-time planning, multi-agent systems, context recognition, vision, strategic decision- making, motor control, intelligent robot control, prediction and many more.

The RoboCup Federation proposes the huge challenge shared by the robotics and artificial intelligence community for next 50 years: *By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup.*

The RoboCup Middle Size League (MSL) is one of the RoboCup robot soccer leagues. In the MSL, two teams of up to 5 robots play soccer on an $18 \times 12m$ indoor field. Each robot is equipped with sensors like cameras and an on-board computer to analyze the current game situation and successfully play soccer. Generally, the rules are the same as the laws of football except for some modifications such as the standard field dimensions or the absence of outsides. The official tournament ball used in matches is any orange FIFA ball. Only 5 robots per team can play on the field, including the goalkeeper.

Until recently, the MSL world was designed to be very simple, the objects of interest being identified by their unique color, like the orange ball, the black obstacles, the green field and the white lines (see Fig. 1.1). In the last few years, fast and robust color segmentation algorithms have been developed to detect and track objects in this scenario in real-time. However, the community agreed that, in the near future, visual cues like color will be removed to come to a more realistic setup with robots playing with a "normal" soccer ball. The color codes tend to disappear as the competition evolves, increasing the difficulty posed to the vision algorithms,

that face a new challenge. The color of the ball, currently orange, is the next color scheduled to become arbitrary. From 2010, inclusive, it will be used an official FIFA ball instead of a ball completely orange.

Solutions are then required for overcoming this new challenge. Efficient methods for detecting and tracking the arbitrary ball in a RoboCup scenario, without the need for color information, should be developed. Most of the approaches to this new difficulty use morphological information.

Morphological object recognition through image analysis has became more robust and accurate in the past years, although still very time consuming, even using modern personal computers. Because RoboCup is a real-time environment, processing time can become a serious constrain when analyzing large amounts of data or executing complex algorithms.

Figure 1.1: Example of a MSL soccer game. This image was acquired during the final game of the MSL at the Robocup 2008 in China: CAMBADA team (with cyan body-markers) playing against Tech United team (with magenta body-markers).

1.2 The CAMBADA team

CAMBADA [8] (acronym of Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture) is the RoboCup Middle Size League soccer team of the University of Aveiro [9], Portugal. This project started officially in October 2003 and, since then, the team has participated in several RoboCup competitions and Portuguese Robotics Festivals, achieving the following results:

- Portuguese Robotics Open 2004: 5th place;
- Portuguese Robotics Open 2005: 4th place;
- Portuguese Robotics Open 2006: 3rd place;
- Portuguese Robotics Open 2007: 1st place;
- RoboCup'2007: 5th place;
- Portuguese Robotics Open 2008: 1st place;
- RoboCup'2008: 1st place;
- Portuguese Robotics Open 2009: 1st place;
- RoboCup'2009: 3rd place.

The team also participated in the following events:

- RoboCup'2004;
- RoboCup'2006;
- DutchOpen'2006.

This project involves people working on several areas for building the mechanical structure of the robot, its hardware architecture and controllers [1, 2] and the software development in areas such as image analysis and processing [3, 4, 5, 6, 7], sensor and information fusion [8, 9],

[8] `http://www.ieeta.pt/atri/cambada`
[9] `http://www.ua.pt`

reasoning and control [10], cooperative sensing approach based on a Real-Time Database [11], communications among robots [12, 13] and process managing [14, 15].

The CAMBADA robots (Fig. 1.2) were designed and completely built in-house. The baseline for robot construction is a cylindrical envelope, with 485mm in diameter. The mechanical structure of the players is layered and modular. Each layer can easily be replaced by an equivalent one. The components in the lower layer, namely motors, wheels, batteries and an electromagnetic kicker, are attached to an aluminum plate placed 8cm above the floor. The second layer contains the control electronics. The third layer contains a laptop computer, at 22.5cm from the floor, an omni-directional vision system, a frontal camera and an electronic compass, all close to the maximum height of 80cm. The players are capable of holonomic motion, based on three omni-directional roller wheels.

Figure 1.2: Robots used by the MSL robotic soccer team CAMBADA.

1.3 Contribution and thesis structure

As from 2010, inclusive, it will be required to use in the MSL competition of RoboCup an arbitrary official FIFA ball instead of a ball completely orange . The robotic soccer team of the University of Aveiro, CAMBADA, wants to solve this problem and migrate its vision system to

a new system that complies to the new requirement. Although the title of this thesis is "*Object detection in robotics using morphological information*", this work focused on the detection of arbitrary ball.

The remaining of the thesis is structured as follows: In Chapter 2, it is presented an overview of the CAMBADA vision system, describing both its hardware and software architectures. The vision system of the other teams in the MSL are also discussed. Chapter 3 addresses the problem of morphological ball detection, including references to previous work done in this area. It is also presented some image processing operations used in the proposed algorithms, namely the Canny edge detector, the Hough Transform and the OpenCV library. The proposed algorithms are described in Chapter 4 and it is divided in the following approaches: specific implementation of the Hough Transform considering a single object, considering multiple objects and Hough Transform using the OpenCV library. After detection, an algorithm is used to validate the objects detected. Results are given in Chapter 5 and Chapter 6 concludes the thesis.

Chapter 2

Vision systems in the MSL

The MSL competition of RoboCup is a standard real-world test for autonomous multi-robot systems. Vision systems are the most important sensing system, providing detailed information about the environment. All teams of the MSL use a camera as its main sensor, deploying the information extraction tasks into the image processing field. The vision system provides fundamental information that is needed for calculating and controlling the behavior of the robotic players.

The vision system should be able to detect objects reliably and provide an accurate representation of the environment around the robot to the higher level processes. The vision system must also be highly efficient, allowing a resource-limited agent to respond quickly to a changing environment. Each frame acquired by a digital camera must be processed in a small, usually fixed, amount of time. Algorithmic complexity is therefore constrained, introducing a trade-off between processing time and the quality of the information extracted.

2.1 Overview of the CAMBADA vision system

In this section, we present a description of the hardware and software architectures of the CAMBADA vision system.

2.1.1 Hardware architecture

The CAMBADA vision system is an hybrid vision system, formed by an omnidirectional vision sub-system and a perspective vision sub-system, that together can analyze the environment around the robots, both at close and long distances.

The omnidirectional vision sub-system [3] is based on a catadioptric configuration implemented with a firewire camera (PointGrey Flea2 camera with a 1/3" CCD sensor and a 4.0mm focal distance lens) and an hyperbolic mirror. This camera can work at 30 fps (frames per second) using the YUV 4:2:2 or RGB modes with a resolution of 640×480 pixels. The perspective vision sub-system uses a low cost firewire front camera (BCL 1.2 Unibrain camera with a 1/4" CCD sensor and a 3.6mm focal distance lens). This camera can deliver 30 fps using the YUV 4:1:1 mode with a resolution of 640×480 pixels.

The information regarding close objects, like white lines of the field, other robots and the ball, are acquired through the omnidirectional sub-system, whereas the perspective sub-system is used to locate other robots and the ball at long distances, which are difficult to detect using the omnidirectional vision system. Thus, the perspective sub-system is very important, for example, for the perception of the environment by the goalkeeper, that needs to see the ball at larger distances.

2.1.2 Software architecture

The software architecture is based on a distributed paradigm, grouping main tasks in different modules. This permits a better understanding of the software work-flow and easier implementation of future improvements. The software can be split in three main modules, namely the *Utility Sub-System*, the *Color Processing Sub-System* and the *Morphological Processing Sub-System*. Each one of these sub-systems labels a domain area where their processes fit, as the case of *Acquire Image* and *Display Image* in the *Utility Sub-System*.

The software architecture used in the omnidirectional and perspective sub-systems is the same, changing only the *Image Mask & Radial Sensors* and the *Distance Mapping Image*.

The *Color Processing Sub-System* includes processes for color classification and extraction, along with an object detection process to extract information, through color analysis, from the acquired image. The vision system is prepared to acquire images in RGB 24-bit, YUV 4:2:2 or

Figure 2.1: The hardware architecture of the vision system developed for the CAMBADA robotic soccer team. On the top, the omnidirectional sub-system with a camera pointing to an hyperbolic mirror. At the bottom of the image, the perspective sub-system with a camera pointing towards the field.

YUV 4:1:1 format using the correct table of conversion. The HSV color space is used for color calibration, due to its special characteristics [6].

The *Morphological Processing Sub-System* includes a color independent ball detection algorithm, more specifically a ball detection algorithm that uses morphological information. The algorithms presented in this thesis are included in this module.

2.2 Vision systems of the other teams

Almost all teams participating in the MSL competition use an omnidirectional camera to acquire information around the robot in all directions. However, some teams also use a perspective sub-system, adding a frontal camera to see objects at larger distances and to complement the vision system - if more information is available, more accurately it can be treated. Furthermore, with two cameras, the vision system is able to use a triangulation technique to calculate the 3D location of the soccer ball.

One difficulty for the teams is the limited processing time, since it is a real-time application. The aim is that the processing time spent on the capture and analysis of images obtained by the

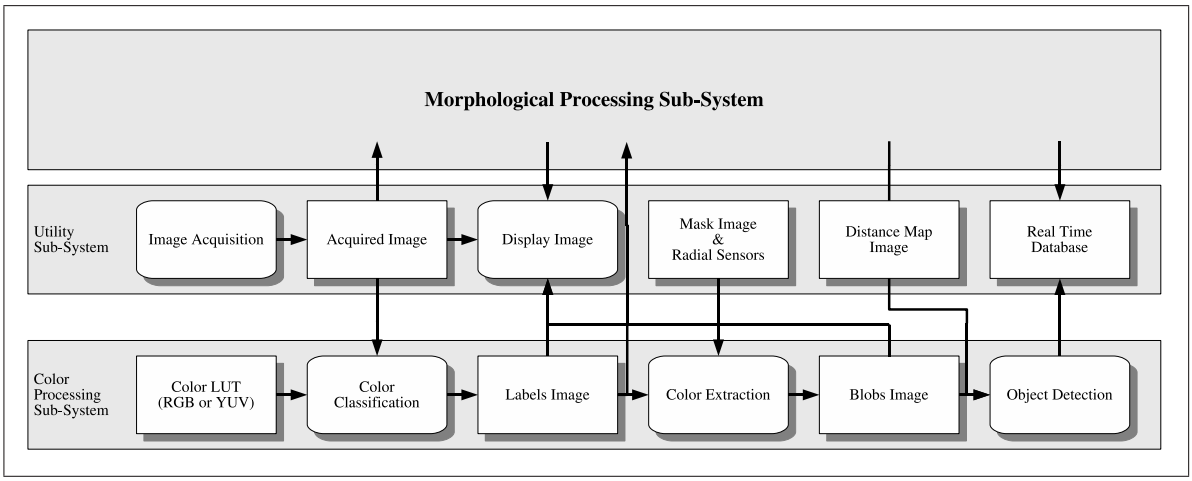


Figure 2.2: The software architecture of the vision system developed for the CAMBADA robotic soccer team.

camera should be the smallest possible. Thus, the high-level algorithms have more processing time and may be more comprehensive and complex. With this idea in mind, some teams are implementing the analysis and processing algorithms of digital images in hardware using dedicated processors such as FPGAs (Field Programmable Gate Array) or DSPs (Digital Signal Processor), instead of running these algorithms on the computer.

In order to be useful, a vision system designed to operate in the MSL competition of RoboCup environment should have:

- image acquisition;
- camera calibration: maps and camera parameters;
- color segmentation;
- object detection: ball, obstacles/teammates and lines.

In the following sub-sections, the vision systems of the most important teams competing in MSL are analyzed. We only considered the teams that have been more innovative and active and, therefore, had better results in the RoboCup competition in recent years.

In terms of arbitrary ball detection, the vision system accuracy of the soccer teams can also be discussed according to the results obtained in the technical challenge denoted “Arbitrary Ball

Challenge"promoted by the RoboCup Federation, where an arbitrary ball has to be found and shot at the goal. This technical challenge has been mandator for all participant teams since 2008. Last year, in the RoboCup'2008, the MRL team won the Challenge and the CAMBADA team attained the second place. This year, the CAMBADA team attained the first place using the algorithms present in this thesis.

2.2.1 Tech United

The Tech United vision system [16] consists of two cameras, an omnidirectional camera at 25 fps and a front camera at 200 fps. This high speed front camera locates balls coming towards the robot with more accuracy, since the omnidirectional vision system is not able to locate balls reaching speeds of 10 meters per second. However, a rate of 200 fps requires a high processing speed, even for a video resolution of 640x480. This difficulty was resolved using a smart camera VC4458, with on-board processing power and high speed, at 200 fps @ 640x480 pixels. Thus, the processing time is greatly reduced because all analysis and processing of the images is done on the camera and only the ball position and velocity is transmitted to the next layer, the low level processing on the main mini-PC.

The Tech United vision system is also able to correct the difference in lighting between frames. The phase and respective frequency of the AC powered light source are detected by running the camera at a high rate for several seconds on a fixed object. The mean intensity of each frame is calculated to join a sine function. This sine function permits to obtain phase and frequency of the illumination source, which is used to trigger the acquired image by the camera to the correct phase.

The fast frontal camera at 200 fps is a gray-scale camera, which permits, with color filtering on the lens, select good candidates for the ball. However, this applies only for the balls with known and fixed color. The ball detection algorithm consists of 4 steps. Initially, it is applied an adaptive threshold; based on the threshold, the regions are labeled; the candidate regions are ordered according to shape descriptors, like roundness or size, calculated for every region; finally, the most likely ball object is selected and its position and size are extracted. The obtained values are then converted to world coordinates and sent by the camera to the main mini-PC.

2.2.2 Brainstormers Tribots

The Brainstormers Tribots vision system [17] consists of two cameras, an omnidirectional camera at 30 fps, used to recognize the ball, the white field markings, the goals, the teammates and the opponents, and a perspective camera, used to recognize the ball from a different point a view. The three dimensional ball position can be calculated by geometric reasoning, using the ball position of each camera. The vertical velocity is estimated by a ball motion estimator component, taking into account gravity and bouncing effects.

Brainstormers Tribots are working on low-level image processing algorithms, specified in Very-high-speed integrated circuits Hardware Description Language (VHDL), directly in hardware, namely using a custom System-on-a-chip (SoC) on FPGA, in order to increase the power efficiency and highly reduce the computer processing time.

2.2.3 1.RFC Stuttgart

The 1.RFC Stuttgart vision system [18] uses stereo vision that consists of two sub-systems: an omnidirectional sub-system with a digital firewire camera at 30 fps and an hyperbolic mirror, and a perspective sub-system with a perspective USB camera also at 30 fps.

The 1.RFC Stuttgart team has been exploring the 3D ball detection with mixed omnidirectional and perspective cameras. This combination, where the omni camera is used for distances of up to 5 meters and a perspective camera for higher distances, is able to detect the ball at distances of up to 14 meters. Thus, up to 8 meters, the ball positions can be fused by geometrical triangulation to obtain a 3D ball position. However, there are some restrictions, such as: low resolution of the cameras, cameras easily decalibrating and the impossibility to use synchronized cameras since the omnidirectional camera operate via firewire and the perspective camera via USB.

In terms of recognition of arbitrary balls, a 2-phase method was developed. During calibration, in phase 1, the image is scanned for circles (the ball) in a predefined region of interest by a generalized symmetry transform and, in phase 2, the color histogram of each circle is extracted. During the game, in phase 1, all circles are obtained by Hough transform and, in phase 2, the color histogram of each circle obtained is extracted to be compared with the color histogram of the calibrated ball.

2.2.4 Hibikino Musashi

The Hibikino Musashi vision system [19] consists of an omnidirectional mirror and a firewire digital camera at 15 fps. Data transmission from the camera to the notebook PC is in YUV format, which is then converted into the HSV format. The vision system is based on YUV and HSV color map spaces, and both color maps have different vector spaces.

The color recognition algorithm is based on the Self-Organizing Map (SOM), which is a learning algorithm that performs a transformation from higher-dimensional vector data spaces to low map spaces. To obtain a robust system against light changing, the recognition of the light color environment and the threshold parameters are both estimated by the SOM.

2.2.5 CarpeNoctem

The CarpeNoctem vision system [20] consists of a firewire omnidirectional camera with a resolution of 640×480 pixels at 30 fps. This camera provides a stream of YUV422-encoded images. A Gain Regulator updates the gain settings of the omnidirectional camera based on estimates of the illumination on the camera lense and on different surrounding areas.

In order to avoid time consuming calibration tasks for color segmentation, almost all calculations are done on gray-scale images, except the ball detection task. A color histogram is calculated from sample images. An approach is used to focus on the most interesting areas to detect the ball by applying a template matching approach on the gradient image. This approach also serves to recognize arbitrary balls.

For estimating the 3D ball position, it is applied a simple multi-hypothesis tracking and it is performed a two-fold sensor fusion approach to combine the information gathered from the omnidirectional and the perspective camera.

2.2.6 NuBot

The NuBot vision system [21] consists of an omnidirectional camera and a perspective camera. In terms of calibration, the Canny edge operator is applied to detect the edges of the panoramic image, where 15 obtained edge points are the support vectors for each predefined direction. Then, a Lagrange interpolation algorithm is used to calculate the distance map of the image. The first step of the interpolation algorithm acts on the radial direction and on each direction

a new support vector is obtained. The second step acts on the rotary direction by using the support vectors obtained in the first step.

To recognize an arbitrary FIFA ball, it is used a method based on the omnidirectional vision system, where the ball can be imaged as an ellipse. The shape information of the ellipse on different locations can be calculated in advance according to the derivation and the calibration result of the distance map. The color variation is scanned to search the possible major and minor axis of the ellipse by radial and rotary scans. Then, it is considered that an ellipse corresponding to the ball may exist if the middle points of the major axis and minor axis are very close to each other in the image. Finally, the ball is verified by matching the color variation points searched before.

A Sobel filter and other techniques are applied to detect all the single-pixel edge points and calculate the gradient directions of these points. Furthermore, the Hough transform algorithm is used to detect the circle imaged by the ball.

2.2.7 MRL

The MRL vision system [22] consists of a firewire omnidirectional camera with a resolution of 659×494 pixels at 70 fps that stands upwards with an hyperboloid mirror above it. This component provides omni-directional vision. The MRL team also implemented a stereo vision system using another camera in front of the goalie to calculate the height of the kicked ball and to increase the precision of recognizing the ball far away from it.

To process the gathered images, at first a median filter is applied in order to reduce image noises, and then the color marks are assigned to each pixel by the Color Lookup Table (CLT). This CLT is used in an image processing algorithm to detect the ball, field, and obstacle areas in the image in real-time at 50 fps on the laptop computer.

To recognize the ball, first, the orange colors are segmented. Then, circular shaped segments are detected enabling the vision system to recognize any standard FIFA ball. For that, it is assigned a coefficient of error to each recognized circle according to how much it is similar to a circle, and the circle with minimum coefficient of error is chosen. Also, there is assumed that black segments in the green area are obstacles. The ball detection is also improved by circle fitting algorithms.

Chapter 3

Morphological ball detection

Some research groups have already started to develop algorithms for color invariant ball detection. Many of the algorithms proposed during previous research work showed their effectiveness but, unfortunately, their processing time is in some cases over one second per video frame. The algorithm proposed by Mitri et al. [23] for learning and detecting soccer balls uses a combination of a biologically inspired computational attention system, calculated in 1.5 seconds approximately, with a fast and complex classifier, calculated in 200 ms.

Hanek et al. [24] proposed a fast Contracting Curve Density (CCD) method for fitting parametric curve models to image data by using local criteria based on local image statistics to separate adjacent regions. The CCD algorithm recognizes the ball without color distribution or specific edge-profile and can extract the contour of the ball even in the presence of strong texture, clutter, partial occlusion and severe changes in illumination. However, the vague position of the ball should be known in advance. Then, the global detection cannot be realized by this approach.

Treptow et al. [25] used the Viola and Jones's algorithm, proposing the first approach to deal with the problem of detecting and tracking objects, namely a soccer ball, in the RoboCup domain, without color information and in real-time, integrating the Adaboost Feature Learning algorithm into a condensation tracking framework.

Mitri et al. [26] presented a scheme for color invariant ball detection, in which the edged filtered images serve as the input of a Gentle Ada Boost learning technique that learns and constructs a cascade of Classification and Regression Trees. This method can detect different soccer balls in different environments, but the false positive rate is high when there are other

round objects in the environment. In contrast to the object detection system proposed by Treptow et al., Mitri et al. pre-process the images by applying an edge detection Sobel filter to enhance the simple vertical and horizontal features. However, this method results in several false detections of all round objects, as well as in undetected balls with lots of background noise or partially visible balls.

Coath et al. [27] proposed an adaptive edge-based arc tracking and arc location scheme to detect a soccer ball in full view and also a ball obscured by a curved object. This system, that processes image data containing edge information, can identify a ball before all of the visible circumference has been traversed, i.e., an early result is possible without processing all significant edge pixels. However, the algorithm is used in a perspective camera vision system in which the field of view is far smaller and the image is also far less complex than that of the omnidirectional vision system used by most of the robotic soccer teams.

More recently, Lu et al. [28] considered that the ball on the field can be approximated by an ellipse. They scan the color variation to search for the possible major and minor axes of the ellipse, using radial and rotary scanning, respectively. A ball is considered if the middle points of a possible major and minor axis are very close to each other in the image. This algorithm doesn't need any learning or training process which is necessary in the recognition algorithm based on Ada Boost learning. It can deal with global detection which is not considered in the Contracting Curve Density algorithm. It is based on an omnidirectional vision system and because of it, the field of view is much wider than those perspective cameras. However, there are some problems in this algorithm, such as: the imaging of the ball is occluded partly by the robot itself when the ball is very close to the robot; the image of the ball cannot be approximated to an ellipse on the image when the ball is imaged by both the horizontally isometric part of the mirror and the vertically isometric part of the mirror partially; and the algorithm can deal only with the situation that the ball is on the field ground. This method has also a processing time that can be 150 ms if the tracking algorithm fails, which might cause problems in real-time applications.

In the remain of the chapter, we will present some image processing concepts, used in the algorithms that will be presented in next chapter.

3.1 Canny edge detector

The Canny edge detector operator [29] is perhaps the most popular edge detection technique. Canny proposed an approach to edge detection that is optimal for step edges corrupted by white noise. In the Canny algorithm, the first derivatives are computed in x and y and then combined into four directional derivatives. The points where these directional derivatives are local maxima are then edge candidates.

The algorithm was formulated with three main criterions:

1. The **detection** criterion expresses the fact that important edges should not be missed and that there should be no spurious responses.

2. The **localization** criterion says that the distance between the actual and located position of the edge should be minimal.

3. The **one response** criterion minimizes multiple responses to a single edge. This is partly covered by the first criterion, since when there are two responses to a single edge, one of them should be considered as false. This third criterion solves the problem of an edge corrupted by noise and works against non-smooth edge operators.

The first requirement aims to reduce the response to noise. This can be effected by optimal smoothing. This algorithm was the first to demonstrate that Gaussian filtering is optimal for edge detection (within his criteria). The second criterion aims for accuracy, i.e., the detected edges appear in the correct position. This can be achieved by a process of non-maximum suppression, which is equivalent to peak detection. Non-maximum suppression retains only those points at the top of a ridge of edge data, while suppressing all others. This results in thinning: the output of non-maximum suppression is thin lines of edge points, in the right place. The third constraint concerns location of a single edge point in response to a change in brightness. This is because more than one edge can be denoted to be present, consistent with the output obtained by earlier edge operators.

The Canny algorithm tries to assemble the individual edge candidate pixels into contours. These contours are formed by applying an hysteresis threshold to the pixels. This means that there are two thresholds, an upper and a lower. If a pixel has a gradient larger than the upper threshold, then it is accepted as an edge pixel; if a pixel is below the lower threshold, it is

rejected. If the gradient of the pixels is between the thresholds, then it will be accepted only if it is connected to a pixel that is above the high threshold. Canny recommended a ratio of high:low threshold between 2:1 and 3:1.

3.2 Hough transform

The Hough transform (HT) [29] is a general technique for identifying the locations and orientations of certain types of features in a digital image, such as straight lines, curves, circles, ellipses (or conic sections) or any particular shape. Developed by Paul Hough in 1962 and patented by IBM, the transform consists of parameterizing a description of a feature at any given location in the original image's space. A mesh in the space defined by these parameters is then generated, and at each mesh point a value is accumulated, indicating how well an object generated by the parameters defined at that point fits the given image. Mesh points that accumulate relatively larger values then describe features that may be projected back onto the image, fitting to some degree the features actually present in the image.

The simplest form of the HT is the Hough line transform, which is a relatively fast way of searching a binary image for straight lines. Its prime advantage is that it can deliver the same result as that for template matching, but faster. This is achieved by a reformulation of the template matching process, based on an evidence gathering approach where the evidence is the votes cast in an accumulator array.

The HT algorithm uses an array, called accumulator, to detect the existence of a line. The dimension of the accumulator is equal to the number of unknown parameters of the Hough transform problem. For example, the linear Hough transform problem has two unknown parameters: a and b. The two dimensions of the accumulator array would correspond to quantized values for the parameters. For each pixel and its neighborhood, the HT algorithm determines if there is enough evidence of an edge at that pixel. If so, it will calculate the parameters of that line, and then look for the accumulator's bin that the parameters fall into, and increase the value of that bin. By finding the bins with the highest values, typically by looking for local maxima in the accumulator space, the most likely lines can be extracted, and their (approximate) geometric definitions read off. The simplest way of finding these peaks is by applying some form of threshold determining which lines are found as well as how many. Since the lines returned do not contain any length information, it is often next necessary to find which parts of the image

match up with which lines. Moreover, due to imperfection errors in the edge detection step, there will usually be errors in the accumulator space, which may make it non-trivial to find the appropriate peaks, and thus the appropriate lines.

The HT can be described as a transformation of a point in the x,y-plane to the parameter space. The parameter space is defined according to the shape of the object of interest. A straight line passing through the points ($x1$,$y1$) and ($x2$,$y2$) can in the x,y-plan be described by

$$y = ax + b.$$

This is the equation for a straight line in the cartesian coordinate system, where a and b represent the parameters of the line. The HT for lines does not uses this representation, since lines perpendicular to the x-axis will have an a-value of infinity. This will force the parameter space a,b to have infinite size. Instead a line is represented by its normal which can be represented by an angle θ and a length ρ.

The parameter space can now be spanned by θ and ρ, where θ will have a finite size, depending on the resolution used for θ. The distance to the line ρ will have a maximum size of two times the diagonal length of the image.

The HT can be used for finding any shape which can be represented by a set of parameters. A circle, for instance, can be transformed into a set of three parameters, representing its center and radius, so that the Hough space becomes three dimensional. Although, the complexity of the transformation increase with the number of parameters needed to describing the shape.

3.2.1 Circular Hough transform

The circle is actually simpler to represent in parameter space, compared to the line, since the parameters of the circle can be directly transferred to the parameter space. The equation of a circle is

$$r^2 = (x - a)^2 + (y - b)^2,$$

Where the point (a, b) is the center of the circle and where r is the radius. The parametric representation of the circle is

$$x = a + r\cos(\theta)$$

$$y = b + r\sin(\theta).$$

Thus, the parameter space for a circle belongs to R^3 (with R the real space). As the number of parameters needed to describe the shape increases as well as the dimension of the parameter space R increases so do the complexity of the Hough transform. In order to simplify the parametric representation of the circle, the radius can be held as a constant or limited to a number of known radii.

Figure 3.1 shows an example of a Circular Hough Transform from the x,y-space (left) to the parameter space (right). This example is for a constant radius.

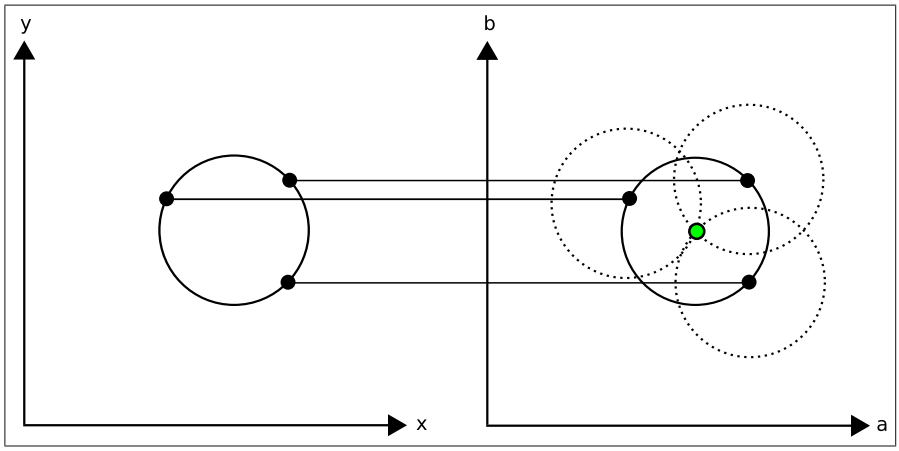


Figure 3.1: A Circular Hough transform example.

Figure 3.2 shows an example of circle detection through the Hough transform. We can see the original image of a dark circle (known radius r) on a white background (see Fig. 3.2a). For each dark pixel, a potential circle-center locus is defined by a circle with radius r and center at that pixel (see Fig. 3.2b) and the frequency with which image pixel occurs in the circle-center loci is determined (see Fig. 3.2c). Finally, the highest-frequency pixel represents the center of the circle (marked by $\bullet$) with radius r (see Fig. 3.2d).

In this way the values are incremented in the accumulator. The accumulator contains numbers corresponding to the number of circles passing through the individual coordinates. Thus, the highest numbers correspond to the center of the circles in the image.

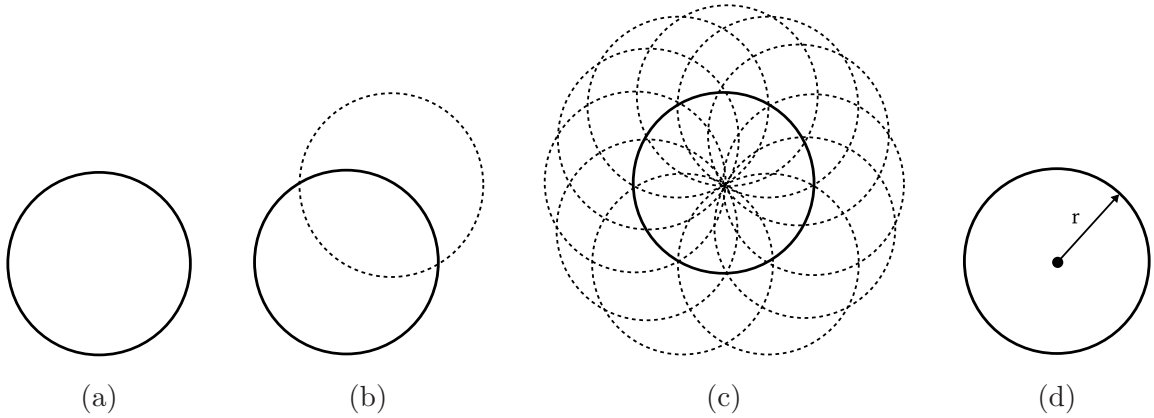


Figure 3.2: Example of circle detection through the Hough transform.

3.3 OpenCV Library

Computer vision is probably the most exciting branch of image processing, and the number of applications in robotics, automation technology and quality control is constantly increasing. The introduction of the OpenCV [1](Open Source Computer Vision Library) is an important milestone addressing system implementation issues in computer vision. Currently it is probably the most widely used vision library for real-time extraction and processing of meaningful data from images.

OpenCV is a computer vision library originally developed by Intel. It is free for commercial and research use under the open source BSD license. OpenCV is written in optimized C and C++ taking advantage of multicore processors and runs under *Linux*, *Windows*, *MacOS X*, *PSP* (PlayStation Portable), *VCRT* (Real-Time OS on Smart camera) and other embedded devices.

OpenCV was designed for computational efficiency and focuses mainly on real-time image processing. The OpenCV library provides more than 500 functions whose performance can be enhanced on the Intel architecture. If available, the Intel integrated performance primitives (IPP) are used for lower-level operations for OpenCV. IPP provides a cross-platform interface to highly optimized low-level functions that perform image processing and computer vision primitive operations. OpenCV also contains a full, general-purpose *Machine Learning Library* (MLL). This sublibrary is focused on statistical pattern recognition and clustering, highly useful for the vision tasks that are at the core of the OpenCV's mission.

[1] `http://sourceforge.net/projects/opencvlibrary`

OpenCV provides the basic tools needed to solve computer vision problems in real-time. In most cases, high-level functionalities in the library are sufficient to solve the more complex problems in computer vision. However, because OpenCV assumes essentially a sequential software architecture, the potential acceleration resources in computer vision are not fully exploited to improve performance.

As already mentioned, the OpenCV library contains over 500 functions that span many areas in vision. In this thesis, we are interested in the field of vision and robotics, more specifically in the processing of images acquired by the robot vision system. Thus, some functions were used, such as [30]:

- void `cvCvtColor` (const CvArr* src, CvArr* dst, int code) - Converts from one color space (a mathematical model to represent colors in digital images) to another while expecting the data type to be the same. The exact conversion operation to be done is specified by the argument code. For example, to convert RGB color space to grayscale, the argument code to use must be `CV_RGB2GRAY`.

- void `cvEqualizeHist` (const CvArr* src, CvArr* dst) - In a standard camera, the shutter and lens aperture settings juggle between exposing the sensors to too much or too little light. Often, the range of contrasts is too much for the sensors to deal with. Hence, there is a trade-off between capturing the dark areas like shadows, which requires a longer exposure time, and the bright areas, which require shorter exposure to avoid saturating whiteouts. After the picture has been taken, there is nothing we can do about what the sensor recorded. However, we can still take what is there and try to expand the dynamic range of the image. Histogram equalization is a method for stretching the range out. In `cvEqualizeHist`, the source and destination must be single-channel, 8-bit images of the same size.

- void `cvSmooth` (const CvArr* src, CvArr* dst, int smoothtype, int param1, int param2, double param3, double param4) - Smoothing or blurring is usually done to reduce noise or camera artifacts. The OpenCV library has five different smoothing operations. All of them are supported through one function, `cvSmooth`, which takes the desired form of smoothing as an argument.

  The `src` and `dst` arguments are the usual source and destination for the smooth operation. The `cvSmooth` function has four parameters with the particularly uninformative names of

param1, param2, param3 and param4. The meaning of these parameters depends on the value of smoothtype, which may take any of the five values CV_BLUR, CV_BLUR_NO_SCALE, CV_MEDIAN, CV_GAUSSIAN or CV_BILATERAL.

Since the Gaussian filter (CV_GAUSSIAN) is the most useful, this filter is used whenever it is necessary to apply smoothing. Gaussian filtering is done by convolving each point in the input array with a Gaussian kernel and then summing to produce the output array. The first two parameters give the width and height of the filter window. The optional third parameter indicates the sigma value (half width at half max) of the Gaussian kernel. If the third parameter is not specified, then the Gaussian will be automatically determined.

- void cvAnd (const CvArr* src1, const CvArr* src2, CvArr* dst, const CvArr* mask) - This function computes a bitwise AND operation on the array src1. Each element of dst is computed as the bitwise AND of the corresponding two elements of src1 and src2. If mask is non-NULL then only the elements of dst corresponding to nonzero entries in mask are computed. Although all data types are supported, src1 and src2 must have the same data type.

- void cvCanny (const CvArr* img, CvArr* edges, double lowThresh, double highThresh, int apertureSize) - The cvCanny function expects an input image img, which must be grayscale, and an output image edges, which must also be grayscale (but which will actually be a Boolean image). The next two arguments, lowThresh and highThresh, are the low and high Canny thresholds, and the last argument is an aperture used by the Sobel derivative operators that are called inside of the implementation of cvCanny.

- CvSeq* cvHoughCircles (CvArr* image, void* circle_storage, int method, double dp, double min_dist, double param1, double param2, int min_radius, int max_radius) - This function implements the circle transform in OpenCV using the Hough gradient method.

  The algorithm is as follows. First, the image is passed through an edge detection phase using cvCanny. Next, for every nonzero point in the edge image, the local gradient is considered (the gradient is computed by first computing the first order Sobel x and y derivatives via cvSobel). Using this gradient, every point along the line indicated by this slope (from a specified minimum to a specified maximum distance) is incremented in the accumulator. At the same time, the location of every one of these nonzero pixels in the edge image is noted. The candidate centers are then selected from those points in this

(two-dimensional) accumulator that are both above some given threshold and larger than all of their immediate neighbors. These candidate centers are sorted in descending order of their accumulator values, so that the centers with the most supporting pixels appear first. Next, for each center, all of the nonzero pixels are considered. These pixels are sorted according to their distance from the center. Working out from the smallest distances to the maximum radius, a single radius is selected that is best supported by the nonzero pixels. A center is kept if it has sufficient support from the nonzero pixels in the edge image and if it is a sufficient distance from any previously selected center.

The input `image` must be an 8-bit image. The `circle_storage` can be a memory storage. If memory storage is used, then the circles will be made into an OpenCV sequence and a pointer to that sequence will be returned. The `method` argument must always be set to `CV_HOUGH_GRADIENT`. The parameter `dp` is the resolution of the accumulator image used. This parameter allows to create an accumulator of a lower resolution than the input image and cannot be less than 1. The parameter `min_dist` is the minimum distance that must exist between two circles in order for the algorithm to consider them distinct circles. The next two arguments, `param1` and `param2`, are the edge Canny threshold and the accumulator threshold, respectively. When `cvCanny` is called internally, the first (higher) threshold is set to the value of `param1` and the second (lower) threshold is set to exactly half that value. The parameter `param2` is the one used to threshold the accumulator. The final two parameters, `min_radius` and `max_radius`, are the minimum and maximum radius of circles that can be found. This means that these are the radii of circles for which the accumulator has a representation.

The result of the function `cvHoughCircles` is returned in the OpenCV sequence `cvSeq`. The OpenCV sequences are an object that can be stored inside a memory storage. Sequences are themselves linked lists of other structures. OpenCV can create sequences out of many different kinds of objects. The sequence can be thought as something similar to the generic container classes (or container class templates). The sequence construct in OpenCV is a *deque* (double-ended queue), so it is very fast for random access and for additions and deletions from either end but a little slow for adding and deleting objects in the middle.

Chapter 4

Proposed algorithm

The proposed algorithm for arbitrary FIFA ball recognition is an algorithm based on the morphological analysis of the acquired image. This algorithm is based on the use of image segmentation, the circular Hough transform and the OpenCV library functions, being strictly directed to detect in the field round objects with specific characteristics, in this case the ball.

The search for potential ball candidates is conducted taking advantage of morphological characteristics of the ball (round shape), using the Hough transform. To feed the Hough transform process, it is necessary a binary image with the edge information of the objects. This image is obtained using the Canny edge detector through the `cvCanny` function. We proposed different solutions that differ in the algorithm used to implement the circular Hough transform and in the process to obtain the image used by the Canny edge detector. The solutions presented can be separated in the following:

- Solution 1: specific implementation of the Hough Transform with color-based image segmentation.
- Solution 2: specific implementation of the Hough Transform without color-based image segmentation.
- Solution 3: Hough Transform using the OpenCV library with color-based image segmentation.
- Solution 4: Hough Transform using the OpenCV library without color-based image segmentation.

We developed an algorithm to search for possible ball candidates through the use of a specific implementation of the Hough Transform (solutions 1 and 2) using some knowledge of the object of interest, in this case the radius of the ball according to the distance to the center of the robot. Since the OpenCV library provides a function dedicated to the search for circles (function `cvHoughCircles`), it was created another version of the algorithms to use this function (solutions 3 and 4).

Regarding the image that will feed the Canny edge detector, the first approach was the development of an algorithm to construct a binary image resulting from the segmentation of the green and non-green regions in the images (solutions 1 and 3). Moreover, to attain a color-independent solution, i.e., a solution in which the image segmentation is not necessary, it was implemented an algorithm (solutions 2 and 4) in which a grayscale image is directly obtained from the acquired image through the OpenCV function, `cvCvtColor`. The results of the proposed solutions are employed in a validation algorithm that is presented later in Section 4.4.

Since the solutions based on the grayscale version of the acquired image (without color-based image segmentation) showed that the edges image is obtained with some noise and with a weak definition, these solutions are not part of the final algorithm.

The main program, `visionTh`, is summarized in Fig. 4.1 and explained next:

- Initially, a lookup table (LUT), used for color segmentation, is created and initialized. The LUT is an array used to replace a runtime computation with a simpler array indexing operation.

- The mask (see Fig. 4.2), an image used to specify the regions of the image that will not be processed (like the body of the robot), is also created and initialized, and to attain a clean image, the edges created by the robot reflection in the omnidirectional mirror are removed.

- The distance map is then created and initialized. It is used to convert image coordinates, in pixels, into real world coordinates, in meters, relative to the center of the robot.

- Then, the sensors (see Fig. 4.3) are created taking into account that parts of the image analysis is based on color search using radial sensors, namely the lines and obstacles detection. The use of radial search lines or radial sensors for object detection accelerates the object detection algorithm without compromising its accuracy, because the acceleration

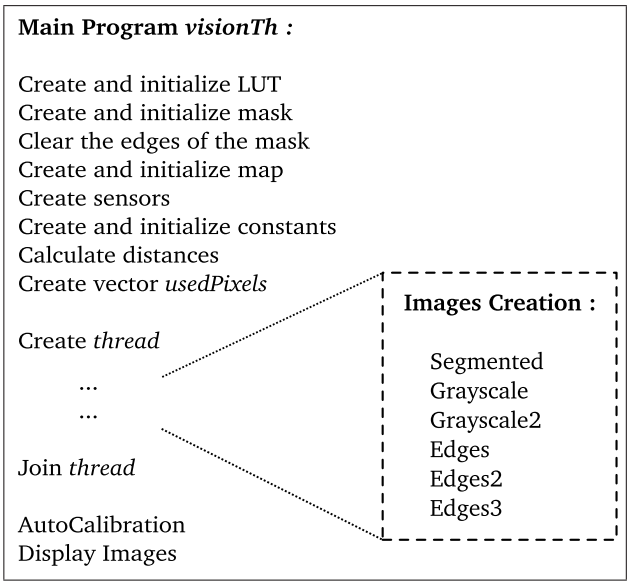


Figure 4.1: The main program `visionTh` with a multi-thread approach.

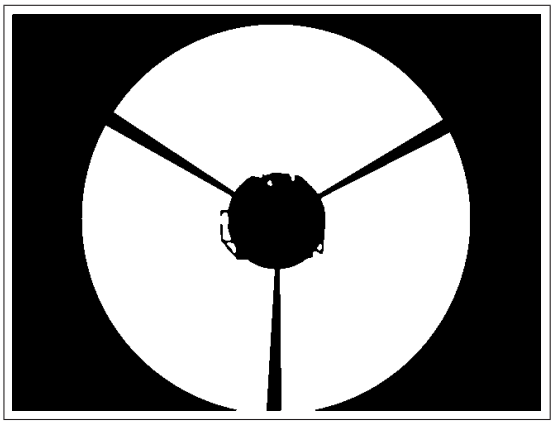


Figure 4.2: Example of mask image containing the valid pixels to be processed from the omnidirectional sub-system. Black pixels mark the pixels that are discarded.

is achieved through the reduction of the search area in the image, since the radial sensors cover only a small percentage of the image [4].

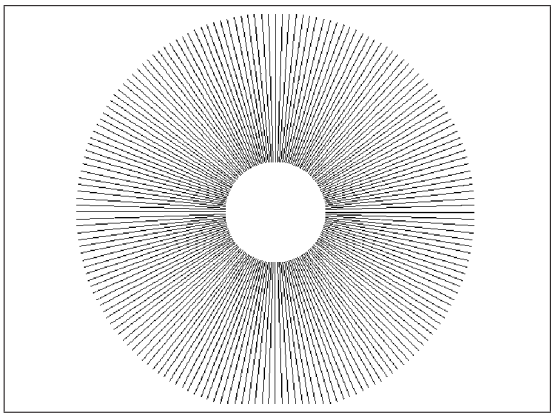


Figure 4.3: The representation of the radial sensors used for color extraction from the omnidirectional sub-system. The radial lines mark the pixels that will be processed.

The creation and initialization of some constants is also performed, such as the vectors `obstaclesDistSeparation`, `obstaclesAngSeparation`, `ballDistSeparation`, `ballAngSeparation`, `ballAngularWidth`, `ballMinNumberPoints` and `ballSquareSize`; the vector that will be most used in the algorithm is the `ballSquareSize` vector and, as the name suggests, it is filled by the sizes of the squares which include the ball; all vectors are filled taking into account the distance to the center of the robot. It also fills a vector with the distances of each pixel of the 640×480 image to the center of the robot. To limit the image to be processed to the pixels of real interest, ensuring a faster processing, are only considered the points within a distance between the robot radius, excluding pixels of the mask. These pixels of real interest are then determined to fill the vector `usedPixels`.

After all these initializations, it is created a thread in order to be possible to perform the most time consuming operations in parallel, taking advantage of the dual core processor used in the laptop of the robots. Since the thread creation to its joining, there are created the images that will be used later (in the next cycle) in the thread processing (see Fig. 4.4). Obviously, during the first cycle of the main program, the images that are used in the thread have not yet been created. The images used in the thread are always determined in the previous cycle of the main program, between the thread creation and its joining. Therefore, the images to be used in the thread have to be copied exactly before the thread creation and have to be cleaned exactly after, because these images are processed in parallel. Precisely after the thread joining,

the images determined in the thread processing must be copied again to be used in the rest of the cycle.

To obtain a good color image, some parameters in the camera must be calibrated, namely exposure, white-balance, gain and brightness. These parameters are calibrated based on the algorithm described in [31].

Finally, the created / calculated / determined images (acquired, segmented, grayscale, edges and Hough) are displayed in the computer, for facilitating the interaction with the user.

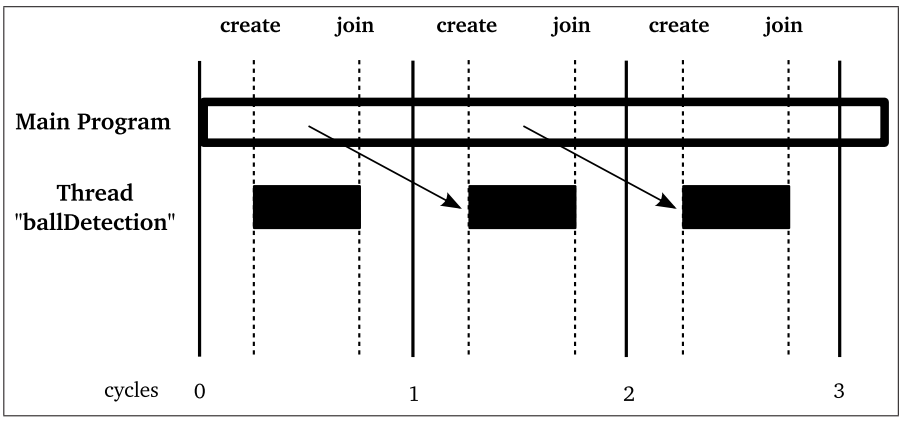


Figure 4.4: Thread `ballDetection`: creation and joining.

Relatively to the images creation (see Fig. 4.5), initially the image is acquired (see Fig. 4.6a) by a digital camera using the omnidirectional sub-system and then it is segmented, creating the `Segmented Image` (see Fig. 4.6b). Next, the `Segmented Image` is used to create the `Grayscale Image` (see Fig. 4.6c). This transformation is based on green pixels (transformed into white pixels) and the non-green pixels (transformed into black pixels). Finally, the `Edges Image` (see Fig. 4.6d) is obtained by applying the Canny edge detector to the `Grayscale Image`.

4.1 Specific implementation of the Hough Transform

One of the approaches for the ball detection was the development of an algorithm using a specific implementation of the Hough Transform. We use some knowledge of the object of interest, in this case the radius of the ball according to the distance to the center of the robot.

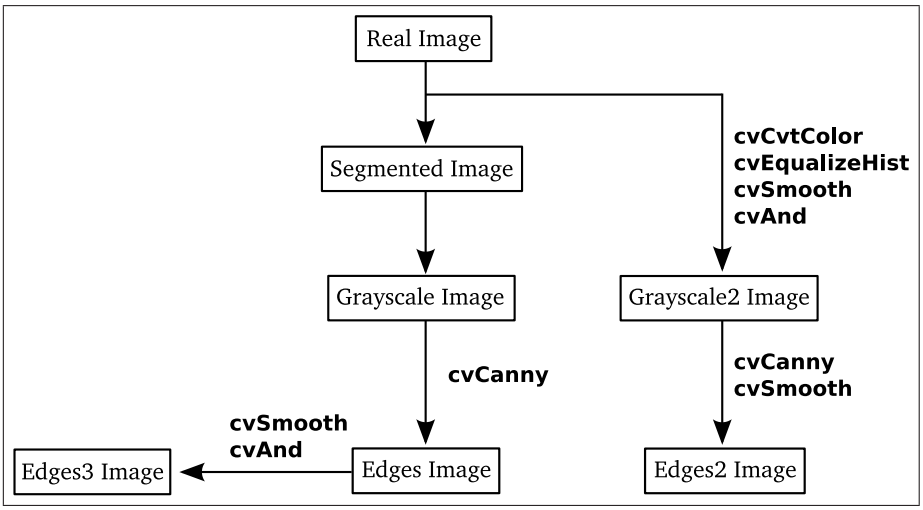


Figure 4.5: Sequence of the images creation.

We know these values a priory, depending on the object and the resolution of the image, and they can be used by the Hough transform to optimize speed. The values were obtained through practical measures, capturing several images by the vision system, with the ball placed at different distances to the center of the robot and with the robot placed static in the center of the field. For each captured image, the distance from the center of the robot to the center of the ball and the ball radius were measured. The measured values were processed to obtain the second-order polynomial displayed in the Fig. 4.7.

4.1.1 Obtaining the absolute maximum value of the Hough Transform

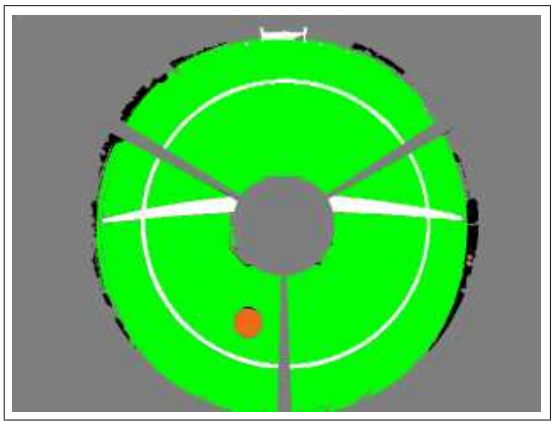
The first approach that we considered was the estimation of the center of the ball considering the pixel given by the position of the absolute maximum value of the Hough Transform.

For all contour pixels obtained after applying the Canny edge detector, we apply the Hough transform according to the example of Fig. 3.2 of the Section 3.2.

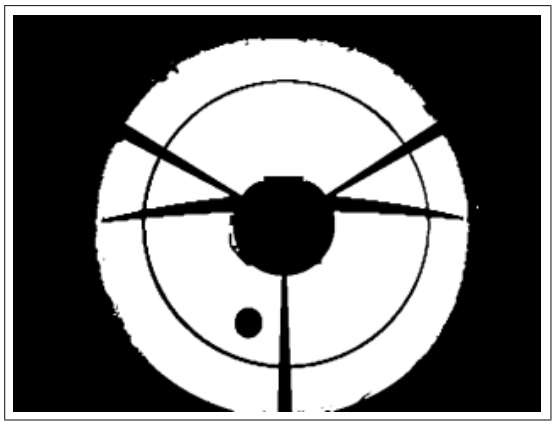
Initially, the `HoughInfo` structure is initialized. The `HoughInfo` structure contains the following variables:

- `value` - the sum of the Hough values;

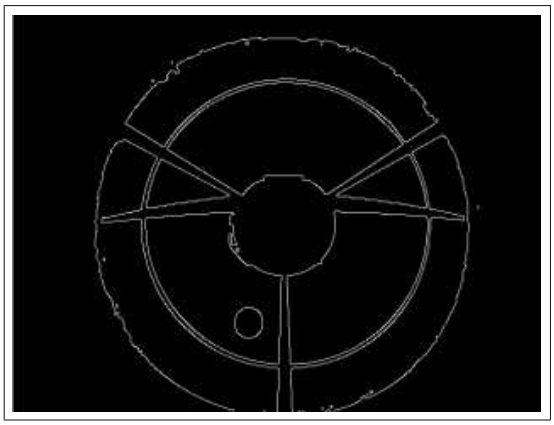
(a) Real Image



(b) Segmented Image



(c) Grayscale Image



(d) Edges Image

Figure 4.6: Sequence of the images transformation. a) `Real Image` acquired by the digital camera, using the omnidirectional sub-system; b) `Segmented Image` that results from the segmentation of the `Real Image`; c) `Grayscale Image` that results from the `Segmented Image`; d) `Edges Image` that results from the Canny edge detector applied to the `Grayscale Image`.

- `point` - coordinates (in pixels) of the point of maximum;
- `dist` - distance of the maximum to the center of the robot (in pixels).

For all the points of the circle associated to the Hough transform, if the point is inside the image and it is not green, the algorithm continues to the next phase. If the distance considered to the center of the robot is less than 70 pixels, then the point of the Hough image is incremented

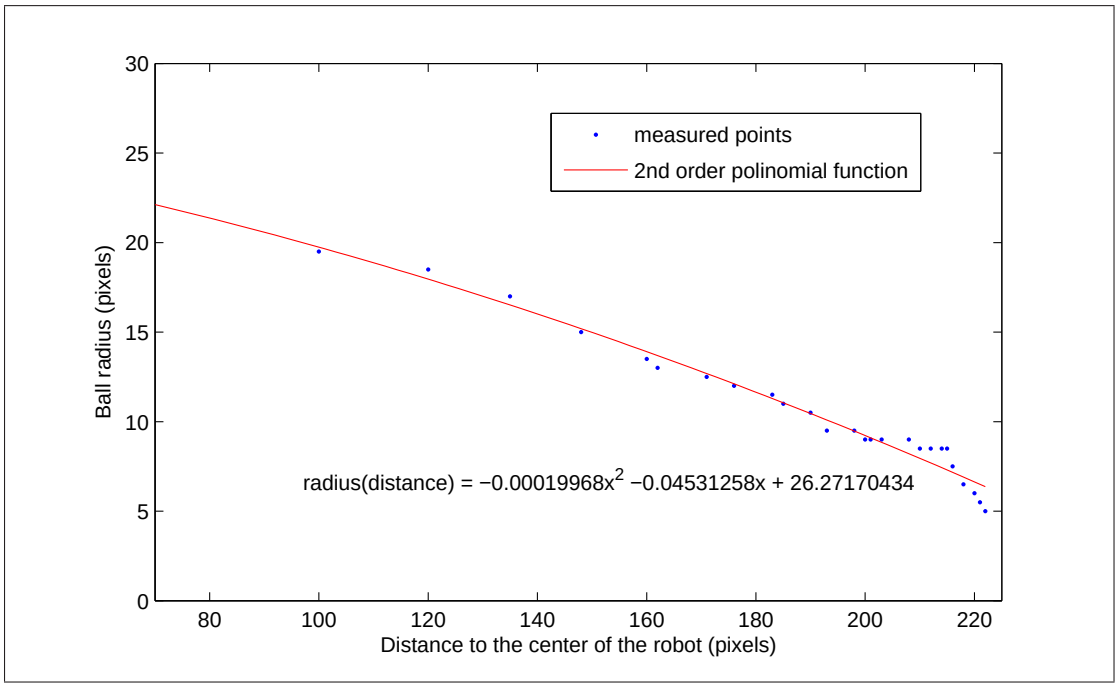


Figure 4.7: Ball radius according to the distance to the center of the robot.

twice. This is due to the fact that the ball is near the robot and is partially occluded. Otherwise, it is incremented only once. The algorithm is outlined in Fig. 4.8.

Using a square with 3×3 pixels, the sum of the Hough Transform is calculated. This value is not more than the sum of the value of the pixel in question with the values of the neighboring pixels. After this calculation, the value of the maximum is obtained.

In the example of Fig. 4.9, there were used four arbitrary balls, as can be seen in the `Edges Image` by their round shapes. Therefore, each ball creates a maximum in the `Hough Image`. However, not only the rounded shapes give rise to maxima, as shown in Fig. 4.9b) and Fig. 4.9c), where it can be seen that the lines of the field and noise, like the presence of human feet, produce false maxima, though with less intensity.

At the end, the ll position is passed to a higher level through the Real-Time Database (RTDB).

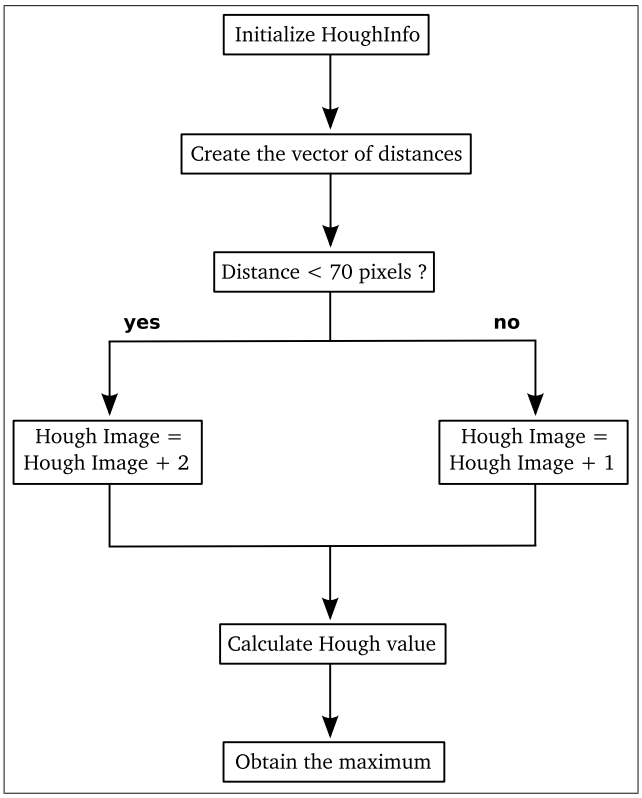
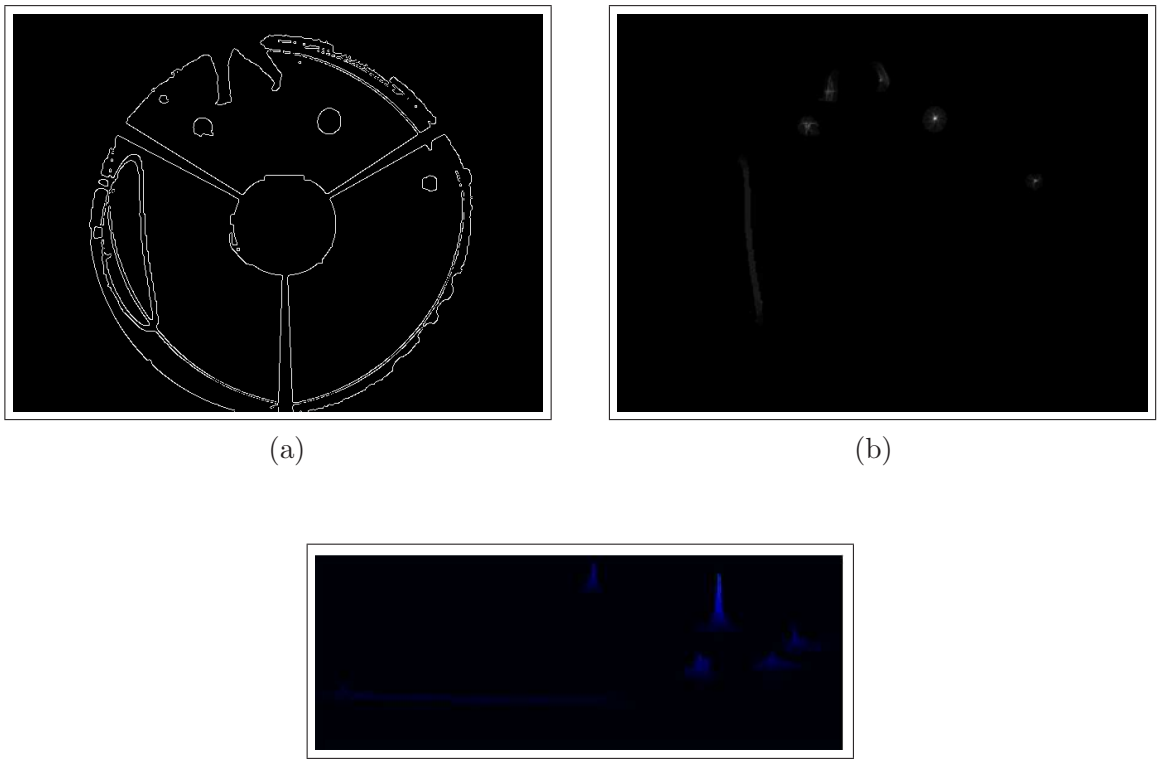


Figure 4.8: Algorithm proposed to obtain the maximum value of the Hough transform.

4.1.2 Multiple maxima

In the previous subsection, it was presented an algorithm that considers the center of the ball as the pixel given by the position of the absolute maximum value of the Hough Transform. However, in some situations, this maximum does not correspond to the ball, mainly when it is at long distances. So, we developed an algorithm to obtain several maximum values, instead of only one, for several distances. Those maximum values are then relative. To perform a more efficient and comprehensive search for those points, there were defined six areas depending on the distance to the center of the robot (see Fig. 4.10). This algorithm (see Fig. 4.11) is then similar to the previous one.

The six areas are described next:



(a) (b)

(c)

Figure 4.9: Illustration of the maxima obtained using 4 balls. a) `Edges Image` that results from the Canny edge detector; b) Result of applying the Hough Transform to the `Edges Image`; c) "3D visualization"of some relative maxima obtained by the Hough Transform.

- Area 1: from 0 until 70 pixels.
- Area 2: from 71 until 105 pixels.
- Area 3: from 106 until 140 pixels.
- Area 4: from 141 until 170 pixels.
- Area 5: from 171 until 205 pixels.
- Area 6: from 206 until 240 pixels.

At the end, the ball position is passed to a higher level through the RTDB.

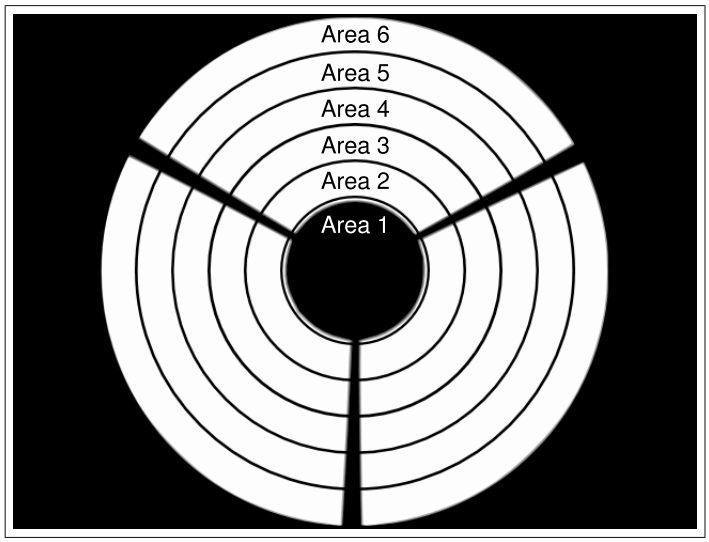


Figure 4.10: Illustration of the six circular areas considered in the image.

4.2 Hough Transform using the OpenCV library

Since the OpenCV library provides a function dedicated to the search for circles, in this section it is presented an algorithm that uses the OpenCV function `cvHoughCircles`. Thus, it was created the function `cvCircles` (see Fig. 4.12) using the referred function. The function `cvHoughCircles` was previously described in Section 3.3.

The input image is the 8-bit single-channel grayscale `Edges3 Image`. The circles are created into an OpenCV sequence and a pointer to that sequence is returned. The method is `CV_HOUGH_GRADIENT` since it is a specification of the OpenCV implementation. The resolution of the accumulator used to detect the centers of the circles is 2, i.e., the accumulator has twice the resolution of the input image and has half width and height. The minimum distance that must exist between two circles, in order for the algorithm to consider them as distinct circles, is 12. This value was obtained by trial and error; it should not be an extreme value because, if it is too small, multiple neighbor circles may be falsely detected in addition to the true one and, if it is too large, some circles may be missed. The higher and lower thresholds used were 180 and 60, respectively. These values were also obtained by testing and according to the Canny recommendation in which the high:low ratio must be between 2:1 and 3:1 (in this case, the ratio

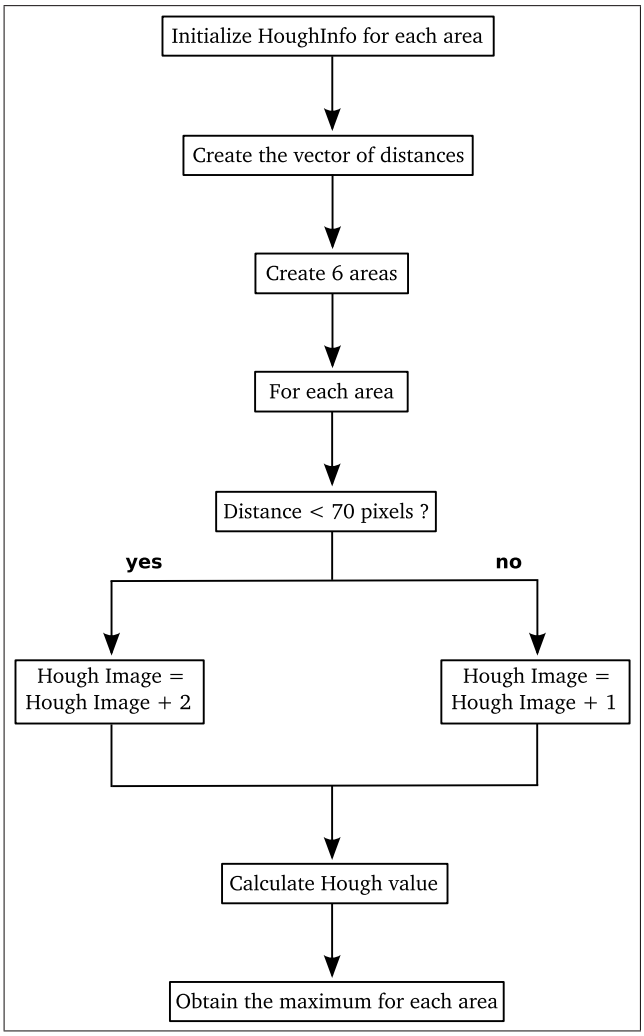


Figure 4.11: Algorithm proposed to obtain the maximum value of the Hough transform, considering the six circular areas presented in Fig. 4.10.

is 180:60 = 3). However, because the `cvCanny` function is called internally, the lower threshold is set to exactly half the higher threshold resulting in a ratio of 2. The minimum radius of circles that can be found is 3 pixels, because for less than 3 pixels it is not considered the existence of ball. The maximum radius of circles that can be found is 25 pixels, because, when the ball is engaged in the robot, the ball radius is approximately 25 pixels.

The `CvCirclesInfo` structure contains the following variables:

- `cvtotal` - number of results that the `cvHoughCircles` function returns;
- `cvx` - x-position in the image;
- `cvy` - y-position in the image;
- `cvradius` - circle radius returned by the `cvHoughCircles` function;
- `cvdist` - distance to the center of the robot, obtained using the x and y positions.

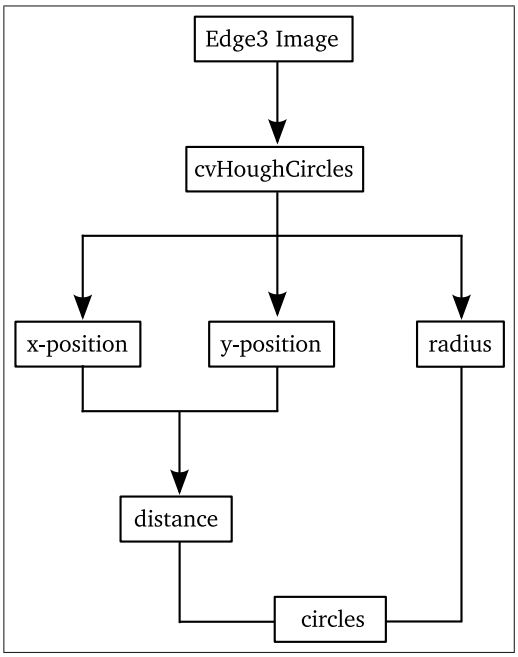


Figure 4.12: Algorithm proposed to search circles using the `cvHoughCircles` function.

At the end, the ball position is passed to a higher level through the RTDB.

4.3 Use of both algorithms

We also tested an algorithm based on the use of the *Specific implementation of the Hough Transform*, followed by the application of the OpenCV function, if the first failed.

In the thread `ballDetection` (see Fig. 4.13), it is called initially the `HoughTransform` function. If this function returns the information that the ball was found, the ball position is calculated through the returned value. The ball position is then corrected taking into account the height of the ball and the mirror. At the end, the ball position is passed to a higher level through the RTDB. However, if the `HoughTransform` returns the information that the ball was not found, the `cvCircles` function is called to perform another test. Then, the expected radius for the distance returned is calculated. If the ball was not found by `cvCircles` or the returned radius is greater than the expected radius, it is sent the information "ball not found" to the higher level, otherwise, the returned value is validated. If the value is considered valid, the ball position is calculated through the returned value, the ball position is corrected and the ball position is passed to the higher level. Otherwise, the information "ball not found" is passed to the higher level.

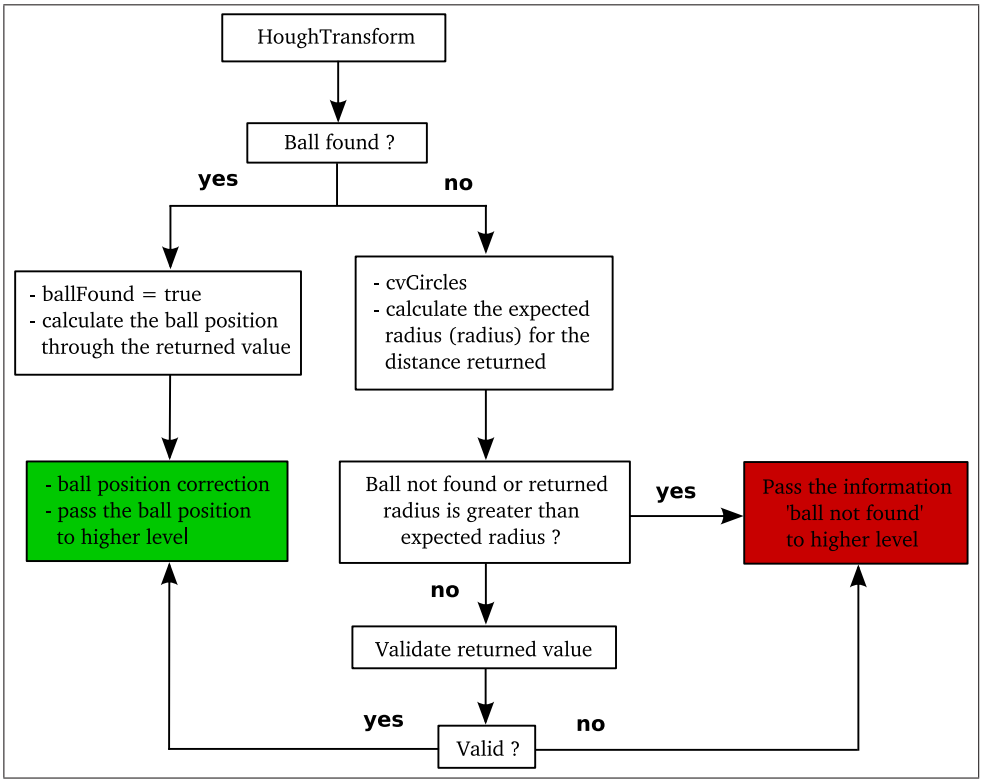


Figure 4.13: Thread `ballDetection` that implements the mixed algorithm.

4.4 Validation

The results of the algorithms described in the previous sections must be validated by a validation process to increase the accuracy of these algorithms.

First, since the ball is never green and knowing that the ball fills almost all the respective square given by the `ballSquareSize` constant, for each distance to the center of the robot, a result will only be considered valid if the percentage of non-green pixels in the square is greater than 85. Based on the same idea, the percentage of non-black pixels must be greater than 50 to eliminate maxima given by other rounded objects which are not balls, like CAMBADA teammates; the assigned value can not be high because there are balls with portions of the black color.

Besides the color validation, it is also performed a validation of the morphology of the candidate, more precisely a circularity validation. Here, from the candidate point to the center of the ball, it is realized a search of pixels at a distance r from the center. For each edge found between the expected radius, plus or minus a defined threshold (after several tests, the threshold chosen was 15%), an edges counter is incremented (see Fig. 4.14). By the size of the square which covers the possible ball and the edges counter, it is calculated the edges percentage.

If the edges percentage is greater than 70, then the circularity of the candidate is verified. When the ball is standing still, the edges percentage is about 90% or more. However, when the ball moves fast through the field, i.e., during real game situations, the motion blur creates a shape less rounded (a nearly elliptical shape) and for avoiding the elimination of the maximum, after several tests, we chose to lower the edges percent needed to validate the candidate to 70%. In Fig. 4.15, it can be seen an example of an `Edges Image` that shows the detection of several maxima. The six relative maxima detected are presented in Fig. 4.16, which were validated by the circularity method, obtaining the results of Table 4.1.

A final result is only validated if the three described validations are verified.

After validating the six relative maxima detected with the circularity validation method, we obtained the values of Table 4.1, where it can be seen that only the fourth maximum (that we know that is the soccer ball) presents an edges percentage higher than 70% (in this case, the edges percentage obtained was 90.2), as it was intended, the circularity validation eliminates some false positives given initially by the Hough transform.

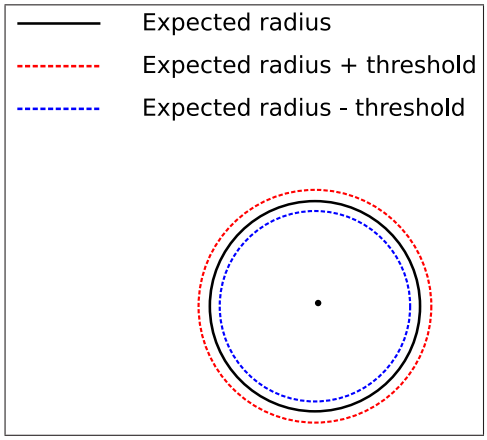


Figure 4.14: Illustration of the threshold used for the circularity validation.

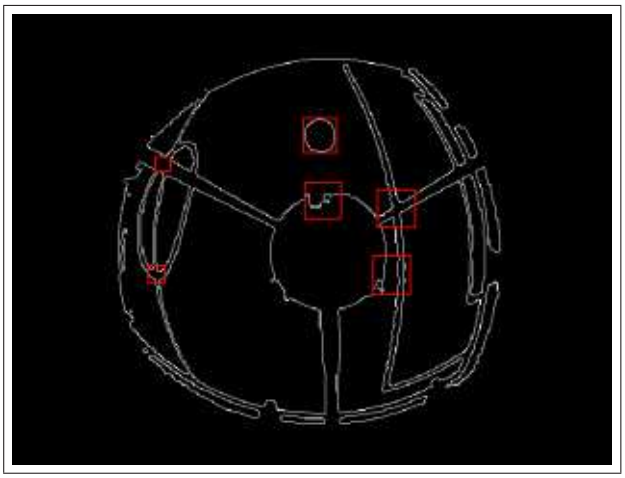


Figure 4.15: Example of an `Edges Image` showing the detection of several maxima.

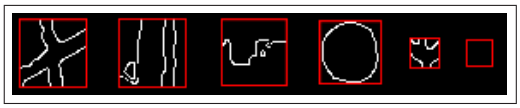


Figure 4.16: The six relative maxima detected for the example of Fig. 4.15.

Maximum	Edges percent (%)
1	6.5
2	10.8
3	15.5
4	90.2
5	27.6
6	0.0

Table 4.1: Edges percentage obtained for the six relative maxima of Fig. 4.16.

Chapter 5

Results

This chapter presents experimental results regarding the algorithms described in the previous chapter. The results have been obtained in the soccer field of the CAMBADA team. In order to evaluate the efficiency of the algorithms developed and to make a comparison between algorithms, it was necessary to test them under the same conditions. All experimental results were obtained by the omnidirectional sub-system. As can be seen in Fig. 5.1, in all these experiments the ball was placed in a measured position in the field (*Real Position*) and the robot has performed a predefined *Tour* while the position of the ball was registered.

Since the aim is to detect arbitrary balls, several various different balls were used in the experiments:

- Ball 1 - A completely orange ball.
- Ball 2 - A mostly black ball.
- Ball 3 - A ball containing 3 colors (white, black and green).
- Ball 4 - A mostly blue ball.
- Ball 5 - A ball containing 2 colors (white and red).

5.1 Specific implementation of the Hough Transform

According to the results of Table 5.1, we have:

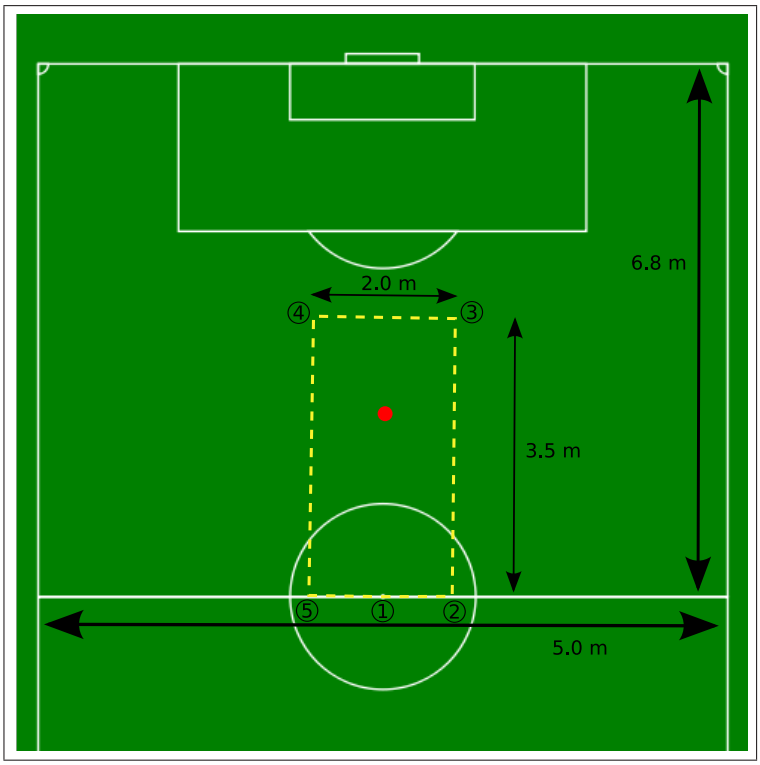


Figure 5.1: Illustration of the predefined *Tour* realized by the robot in the CAMBADA field. The robot starts at the point 1, passes through the points 2, 3, 4 and 5, and returns to the point 1 again.

- In the tour 1, it is noted that the ball was always detected as shown by the value of *Detection ratio*. The low value of standard deviation (*std*) obtained indicates little scatter and high accuracy. The results show a good effectiveness of the method implemented. However, the ball used was a completely orange ball, allowing a good segmentation of the ball in the real image, resulting in almost perfectly circular edges.

- In the tour 2, the value of *Detection ratio* is significantly above the previous case, but the value of the *std* is smaller, showing that the ball was detected less often, but the detection was more precise. It should also be noted that these results are due to the fact that the color of the ball allows a segmentation almost as good as the segmentation of the completely orange ball, although the percentage of ball detection has been less than expected, given

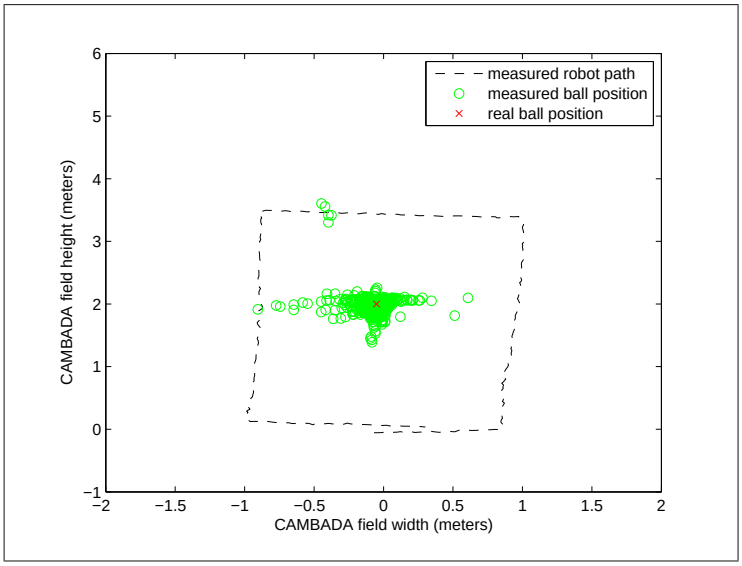


Figure 5.2: Tour 1 - Results using the algorithm with the specific implementation of the Hough Transform, using the ball 1.

Robot Tour	Ball Used	Real Position	Measures		
			Average	Std	Detection ratio (%)
1	1	(0.05, 2.00)	(0.03 , 1.92)	(0.61 , 0.56)	100.0
2	2	(-0.10 , 1.95)	(-0.09 , 1.94)	(0.24 , 0.10)	84.7
3	3	(-0.05 , 2.10)	(-0.12 , 2.09)	(0.21 , 0.27)	100.0
4	4	(-0.05 , 1.95)	(0.38 , 2.01)	(1.29 , 0.25)	97.6
5	5	(-0.05 , 2.15)	(0.02 , 2.25)	(0.63 , 0.22)	94.7

Table 5.1: Some measures obtained for the experiments presented in Fig. 4.11. All the measures are in meters, except for *Detection ratio* that shows the percentage of samples collected where the ball was detected.

the value of the previous experience.

- In the tour 3, it is noted that the ball was always detected as shown by the value of *Detection ratio*. With this method, the best results were obtained in this experience.

- This method was the worst, in tour 4, although a sufficiently low value of the *std* was

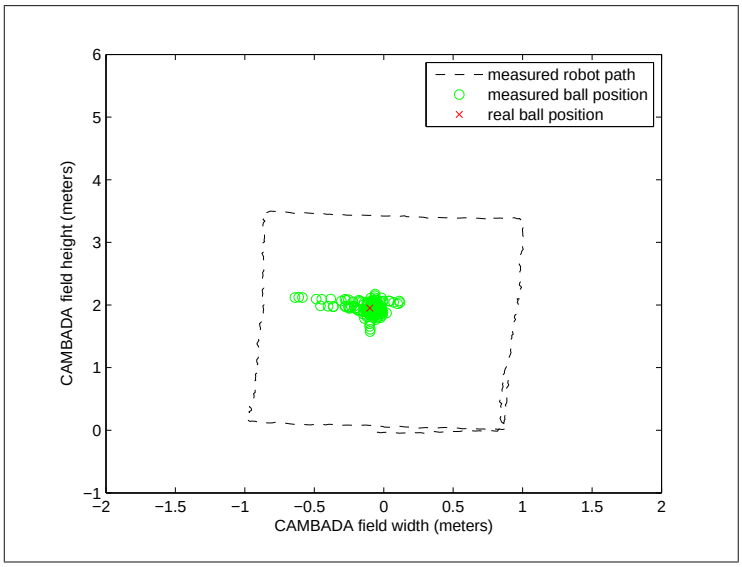


Figure 5.3: Tour 2 - Results using the algorithm with the specific implementation of the Hough Transform, using the ball 2.

obtained. Using this method, this was the largest value of *std* obtained, although it is low enough to consider a fairly accurate ball detection.

- In the tour 5, the value of the *std* obtained is sufficiently low and the *Detection ratio* is still acceptable.

As can be seen in the charts of Figs. 5.2 - 5.6, and in Table 5.1, the algorithm implemented provides frequent, but not always accurate, ball detection.

5.2 Hough Transform using the OpenCV library

According to the results of Table 5.2, we have:

- In the tour 6, it is noted that the ball was always detected, as shown by the value of *Detection ratio*. The low value of *std* obtained indicates high accuracy. The results show a good effectiveness of the method implemented. However, as explained before, the ball used

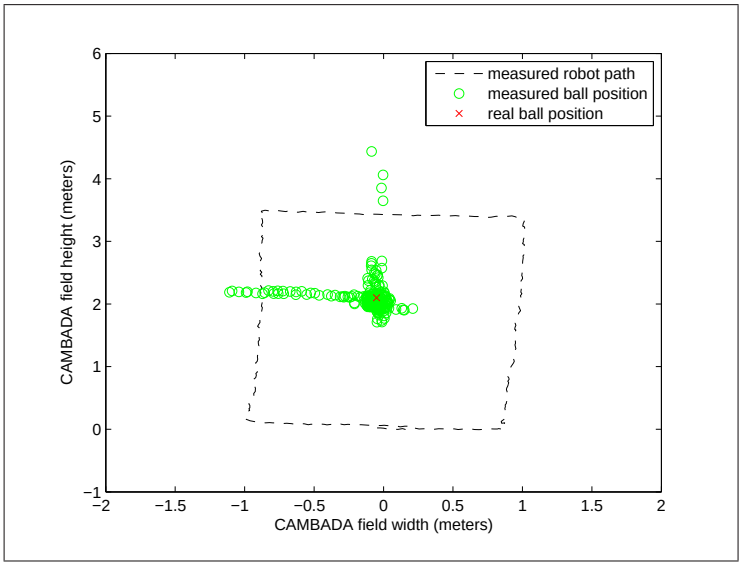


Figure 5.4: Tour 3 - Results using the algorithm with the specific implementation of the Hough Transform, using the ball 3.

Robot Tour	Ball Used	Real Position	Measures		
			Average	Std	Detection ratio (%)
6	1	(-0.05 , 2.00)	(-0.02 , 1.96)	(0.07 , 0.07)	99.7
7	2	(-0.05 , 2.05)	(-0.05 , 2.13)	(0.07 , 0.07)	92.3
8	3	(-0.05 , 2.00)	(-0.03 , 2.05)	(0.06 , 0.10)	50.8
9	4	(-0.05 , 2.00)	(-0.07 , 2.03)	(0.08 , 0.05)	36.2
10	5	(0.00 , 2.20)	(0.01 , 2.24)	(0.05 , 0.07)	74.5

Table 5.2: Some measures obtained for the experiments presented in Fig. 4.12. All the measures are in meters, except for the *Detection ratio* that shows the percentage of samples collected where the ball was detected.

was a completely orange ball, allowing a good segmentation of the ball in the real image, resulting in almost perfectly circular edges.

- In the tour 7, the value of the *std* obtained was the same as the value of the previous experience.

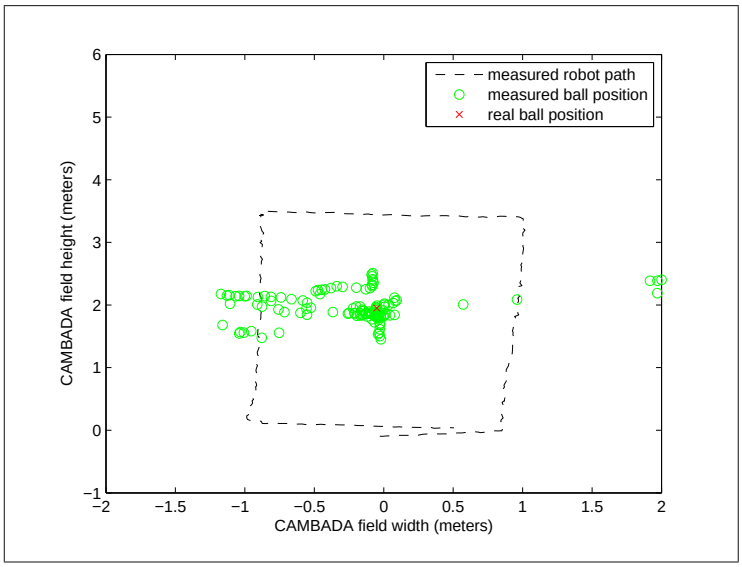


Figure 5.5: Tour 4 - Results using the algorithm with the specific implementation of the Hough Transform, using the ball 4.

- In the tour 8, although the low value of *std* obtained indicates high accuracy, the ball was detected only approximately 50% of the time, as shown by the value of *Detection ratio*.
- Using this method, the lowest value of *Detection ratio* obtained was in tour 9. The low value of *std* indicates again high accuracy.
- In the tour 10, using this method, we got the best value of the *std*, showing a very accurate ball detection. Although the *Detection ratio* obtained is higher than the *Detection ratio* of the two previous experiments, the value is still low to consider an accurate ball detection.

As can be seen in the charts of Figs. 5.7 - 5.11, and in Table 5.2, the algorithm implemented results in intermittent but very precise ball detection.

5.3 Use of both algorithms

According to the results of Table 5.3, we have:

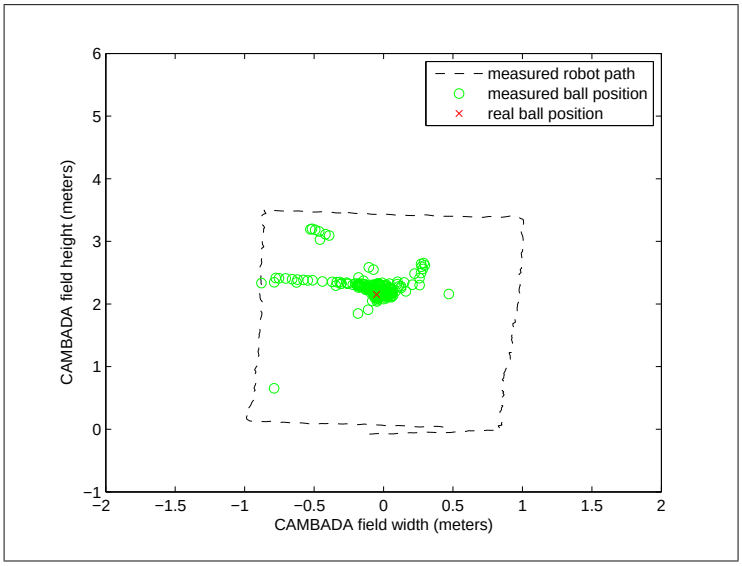


Figure 5.6: Tour 5 - Results using the algorithm with the specific implementation of the Hough Transform, using the ball 5.

Robot Tour	Ball Used	Real Position	Measures		
			Average	Std	Detection ratio (%)
11	2	(-0.05 , 2.00)	(0.01 , 1.77)	(0.47 , 1.08)	93.4
12	3	(-0.05 , 2.00)	(-0.06 , 1.96)	(0.07 , 0.11)	93.9
13	4	(-0.05 , 2.10)	(-0.05 , 2.20)	(0.07 , 0.05)	91.4
14	5	(-0.05 , 2.00)	(-0.11 , 1.95)	(0.32 , 0.32)	99.6

Table 5.3: Some measures obtained for the experiments presented in Fig. 4.13. All the measures are in meters, except for the *Detection ratio* that shows the percentage of samples collected where the ball was detected.

- In the tour 11, the value of the *std* was worse than the previous algorithms, and the value of *Detection ratio* is acceptable.
- In the tour 12, the low value of the *std* indicates high accuracy and the *Detection ratio* is still acceptable.
- In the tour 13, we obtained the lowest value of the *Detection ratio*, although still acceptable.

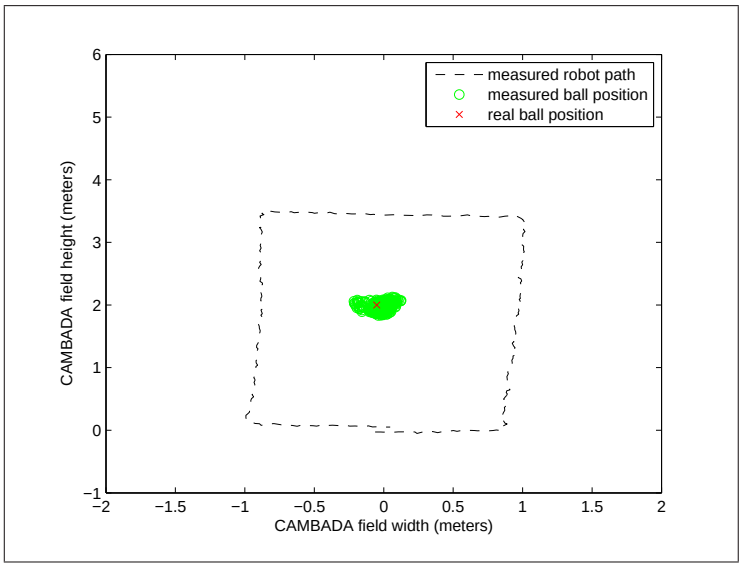


Figure 5.7: Tour 6 - Results of the Hough Transform using the OpenCV library, using the ball 1.

Using this method, we got the best value of the *std*, showing a very accurate ball detection.

- In the tour 14, it is noted that the ball was always detected as shown by the value of the *Detection ratio* and the value of the *std* obtained is sufficiently low to consider a good precision.

As can be seen in the charts of Figs. 5.12 - 5.15 and in Table 5.3, the algorithm implemented provides good and very precise ball detection, i.e., these experiments showed that the use of both implemented algorithms results in a more accurate and effective ball detection.

Some of the missings of the correct position are also due to some other parts / components of the robot software, namely the sensor fusion and integration process and the errors in the mapping that converts image coordinates, in pixels, into real world coordinates, in meters, relative to the center of the robot.

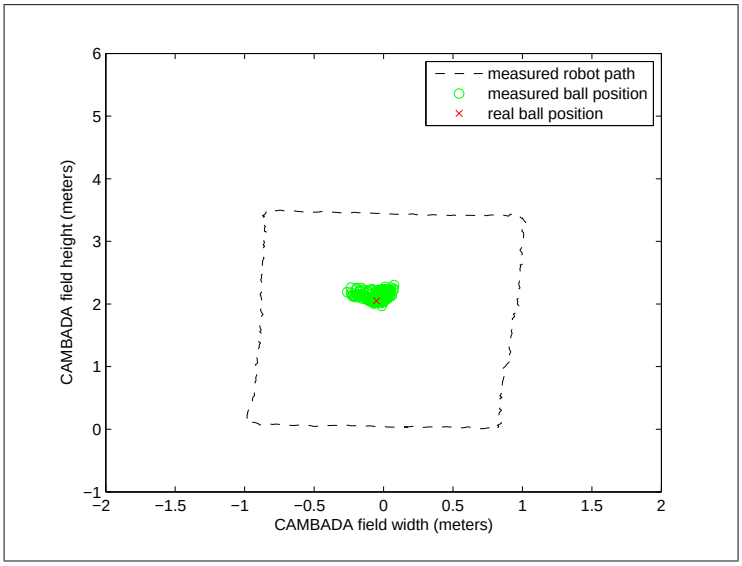


Figure 5.8: Tour 7 - Results of the Hough Transform using the OpenCV library, using the ball 2.

5.4 Processing Times

In this section we present the most important processing times. The average processing times obtained to segment each acquired video frame (creating the `Segmented Image`), and to create the edges (creating the `Edges Image`) were 6 ms and 5 ms, respectively. To obtain the total processing time for each frame, these values must be added to the processing times of Table 5.4, where are shown the processing times for each algorithm.

Each frame acquired must be processed in a small and fixed amount of time. In this case, since the camera used by the CAMBADA omnidirectional vision system acquires images at 30 fps, the time available for processing the acquired image is 1/30 fps = 33.3 ms.

According to the results of Table 5.4, we have that the available processing time (33.3 ms) allows the algorithm 1 is to be completely performed, but the same cannot be guaranteed for the other two algorithms (algorithms 2 and 3).

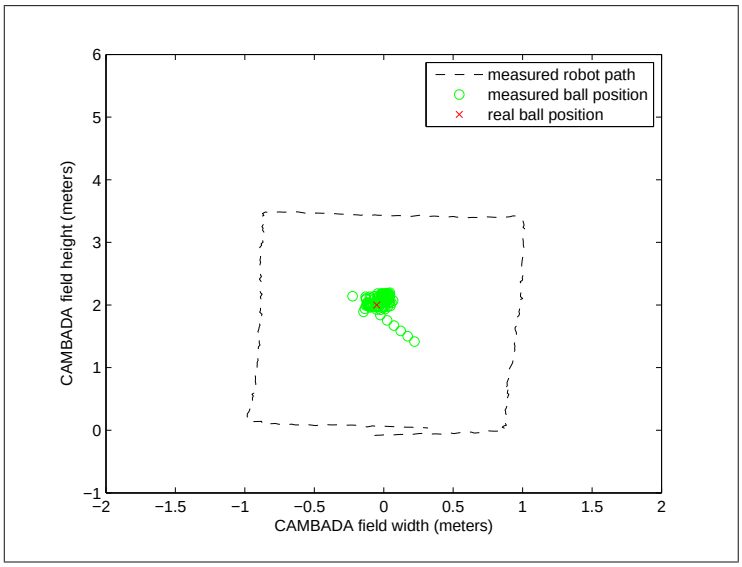


Figure 5.9: Tour 8 - Results of the Hough Transform using the OpenCV library, using the ball 3.

Algorithm Used	Processing times (ms)		
	Minimum	Maximum	Average
1	1	10	5.2
2	18	46	23.5
3	4	48	19.9

Table 5.4: Processing times for each implemented algorithm: 1 - *Specific implementation of the Hough Transform*; 2 - *Hough Transform using the OpenCV library*; 3 - *Use of both algorithms.*

5.5 Results obtained in the RoboCup 2009

The CAMBADA team participated in the technical challenge called "Arbitrary Ball Challenge", in the RoboCup 2009. This challenge is carried out with three different standard FIFA balls. A robot is placed on the field and the ball is placed in front of the robot for 5 seconds. Afterwards the ball is placed at an arbitrary position on the field. Immediately after, the robot has 60 seconds to find the ball and to dribble it into a predefined goal. One point is awarded to the robot for correctly identifying the ball, i.e., the robot has found and touched the

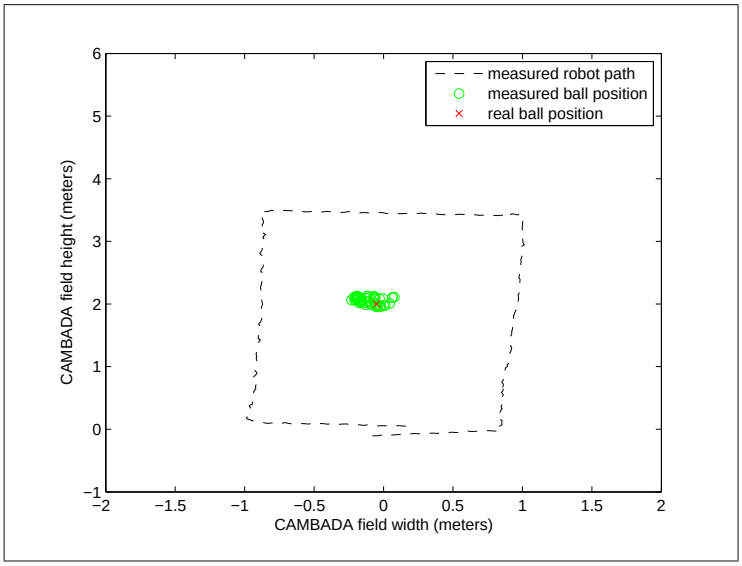


Figure 5.10: Tour 9 - Results of the Hough Transform using the OpenCV library, using the ball 4.

ball for the first time. A second point is awarded if the robot has scored a goal. In total this challenge is repeated three times with varying balls but always with the same robot. In total a team can be awarded up to six points for this challenge.

The robot used by the CAMBADA team correctly identified the ball twice (see Figs. 5.16a and 5.16c) and shot the ball at the goal (see Fig. 5.16b), achieving four points. Unfortunately, the robot could not identify the third ball, probably because the head of the robot, due to the type of illumination of the field, created a circular shaped shadow in the midfield area, which became a ball candidate for the robot.

In both cases, the robot identified the ball and dribbled it into a goal in, approximately, 20 seconds.

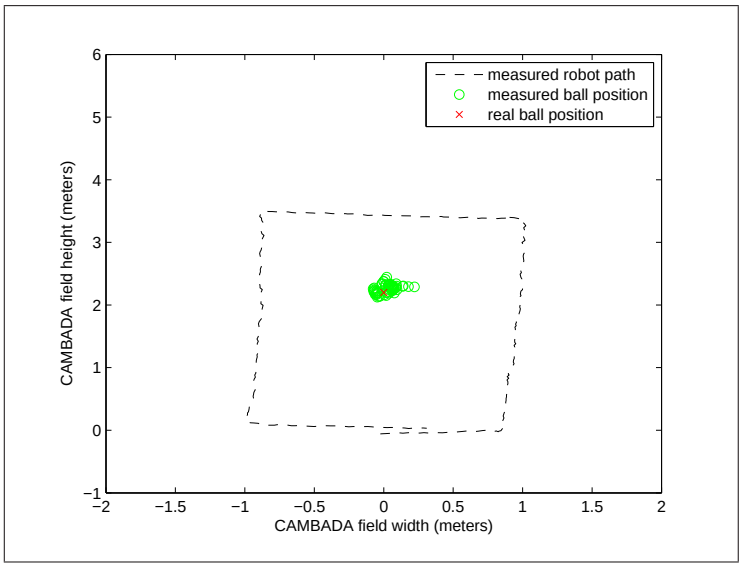


Figure 5.11: Tour 10 - Results of the Hough Transform using the OpenCV library, using the ball 5.

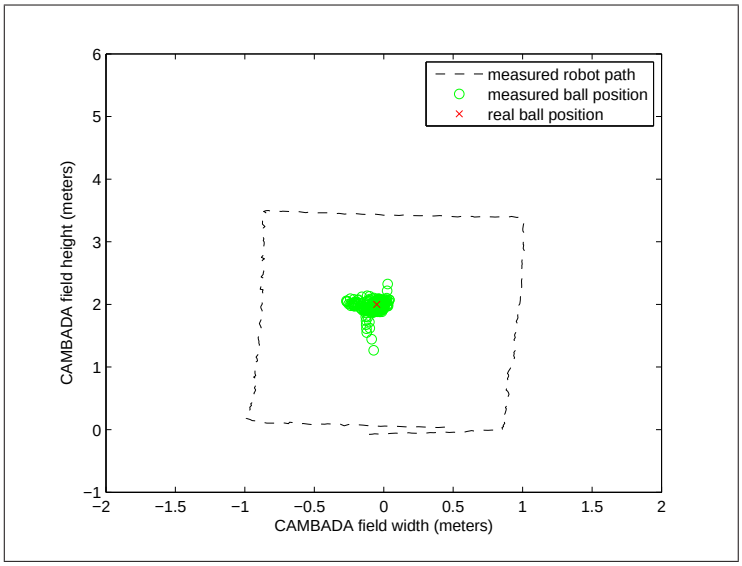


Figure 5.12: Tour 11 - Results obtained after the use of both algorithms, using the ball 2.

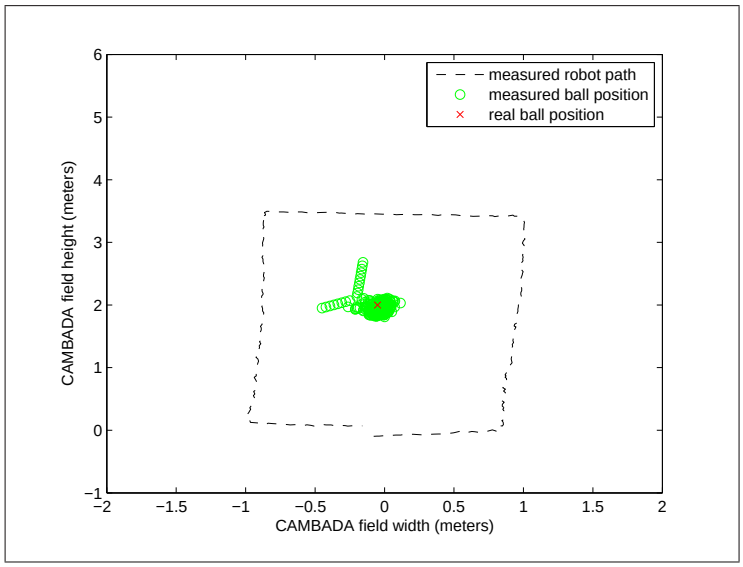


Figure 5.13: Tour 12 - Results obtained after the use of both algorithms, using the ball 3.

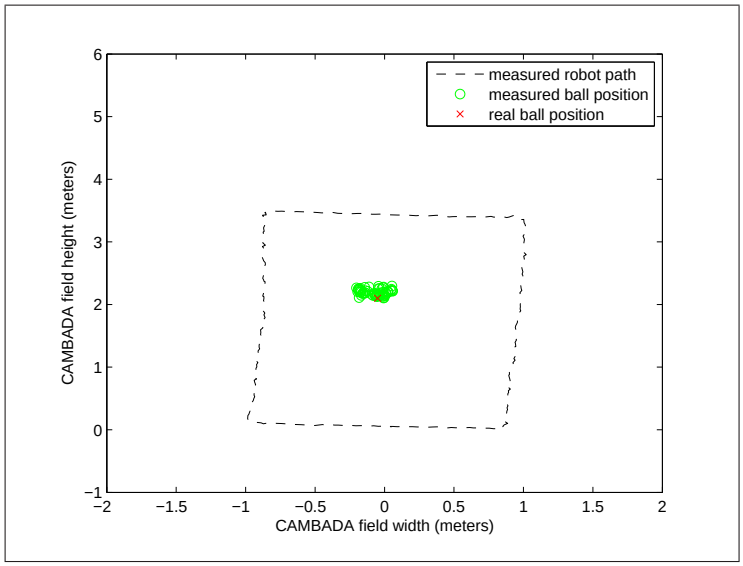


Figure 5.14: Tour 13 - Results obtained after the use of both algorithms, using the ball 4.

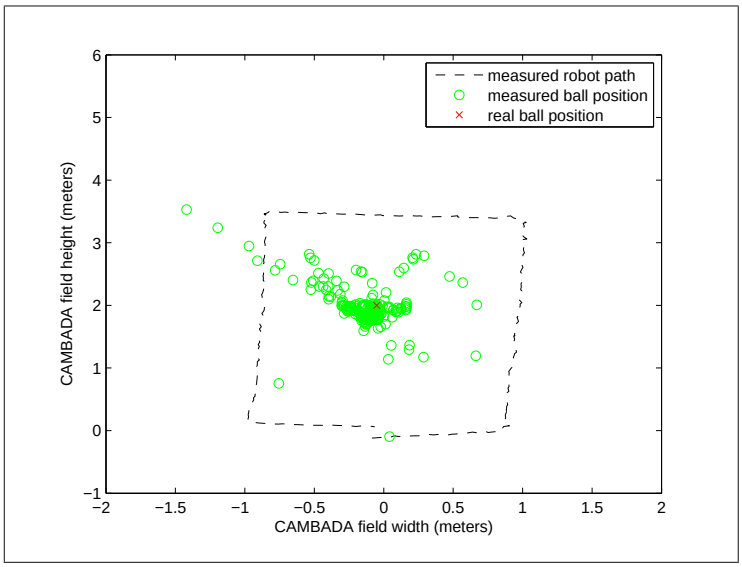


Figure 5.15: Tour 14 - Results obtained after the use of both algorithms, using the ball 5.

(a) The robot finds the first ball.

(b) The robot shot the ball at the goal.

(c) The robot finds the second ball.

Figure 5.16: Participation of the CAMBADA team in the "Arbitrary Ball Challenge", in the RoboCup 2009.

Chapter 6

Conclusions

In this work, we addressed the field of real-time robotic vision, using the example of the CAMBADA vision system. The work had as main objective the development of an efficient vision system for an autonomous robot, designed to play soccer football in the MSL competition of RoboCup. In particular, we developed algorithms for the detection of arbitrary FIFA balls, an important object for soccer robots. All the algorithms developed were implemented in the C++ language, yet the most popular language for vision system implementation.

The algorithms were designed using two approaches. On one hand, the image being processed by the Hough transform was obtained using color information, i.e., the grayscale image used to obtain the edges image is obtained using the acquired image segmentation and some filtering. On the other hand, the image was obtained using the direct conversion between the acquired image in RGB and the grayscale image, through the OpenCV function `cvCvtColor`. Since the solutions based on this approach showed that the edges image is obtained with more noise and with little definition, these solutions are not part of the implemented algorithm.

Then, we developed three algorithms in order to detect arbitrary soccer balls. The first, named "specific implementation of Hough Transform ", is an algorithm where the Hough Image is created through the drawing of circles and the center of the circle is probably the point of highest value, which meets certain conditions. The second, named "Hough Transform using the OpenCV library ", is an algorithm where the Hough transform is performed using an existing implementation, more precisely using the OpenCV function `cvHoughCircles`, created specifically for circle detection, utilizing the fact that the ball presents a rounded shape, almost circular.

The last algorithm is the merge of the two previous algorithms; the first algorithm is performed, and if it does not get any result, it performes the second, allowing a more effective ball detection. The ball candidates obtained by these algorithms are validated by a validation process.

Given the results of Chapter 5, it can be concluded that:

- First, since the detection is made by morphological analysis, and taking into account that the analyzed contours are obtained using the acquired image segmentation, the shape of the ball depends on the color image. The more defined is the color image segmentation, closer to a circle is the form obtained. Therefore, the results in the completely orange ball detection were better than the results obtained in the detection of other balls.

- In terms of *Detection ratio*, the "*Specific implementation of the Hough Transform*" algorithm has shown to be better than the "*Hough Transform using the OpenCV library*" algorithm and, in terms of *Standard Deviation* the opposite happens, i.e., with the first method, the ball is detected more often but with the second the ball detection is more precise.

- The use of both implemented algorithms results in a more accurate and effective ball detection algorithm.

- The *Hough Transform using the OpenCV library* algorithm resulted in intermittent but very precise ball detection. This intermittency can be explained by the fact that the processing time available (33.3 ms) is exceeded by the time required ($23.5 + 6 + 5 = 34.5ms$). The same applies to the *Use of both algorithms* algorithm, where the maximum processing time obtained was 48 ms.

Globally, the experimental results are promising. Moreover, the CAMBADA team achieved the 1st place in the technical challenge called "Arbitrary Ball Challenge", where these algorithms were tested in real applications. These results are a prove of the effectiveness of the proposed and accomplished work.

Bibliography

[1] J. L. Azevedo, B. Cunha, and L. Almeida. Hierarchical distributed architectures for autonomous mobile robots: a case study. In *Proc. of the 12th IEEE Conference on Emerging Technologies and Factory Automation, ETFA2007*, pages 973–980, Greece, 2007.

[2] L. Almeida, P. Pedreiras, and J. A. Fonseca. The FTT-CAN protocol: Why and how. *IEEE Transactions on Industrial Electronics*, 49(6):1189–1201, 2002.

[3] A. J. R. Neves, G. Corrente, and A. J. Pinho. An omnidirectional vision system for soccer robots. In *Progress in Artificial Intelligence*, volume 4874 of *Lecture Notes in Artificial Inteligence*, pages 499–507. Springer, 2007.

[4] A. J. R. Neves, D. A. Martins, and A. J. Pinho. A hybrid vision system for soccer robots using radial search lines. In *Proc. of the 8th Conference on Autonomous Robot Systems and Competitions, Portuguese Robotics Open - ROBOTICA'2008*, pages 51–55, Aveiro, Portugal, April 2008.

[5] D. A. Martins, A. J. R. Neves, and A. J. Pinho. Real-time generic ball recognition in RoboCup domain. In *Proc. of the 11th edition of the Ibero-American Conference on Artificial Intelligence, IBERAMIA 2008*, Lisbon, Portugal, October 2008.

[6] P. M. R. Caleiro, A. J. R. Neves, and A. J. Pinho. Color-spaces and color segmentation for real-time object recognition in robotic applications. *Revista do DETUA*, 4(8):940–945, June 2007.

[7] B. Cunha, J. L. Azevedo, N. Lau, and L. Almeida. Obtaining the inverse distance map from a non-SVP hyperbolic catadioptric robotic vision system. In *Proc. of the RoboCup 2007*, Atlanta, USA, 2007.

[8] J. Silva, N. Lau, J. Rodrigues, and J. L. Azevedo. Ball sensor fusion and ball interception behaviours for a robotic soccer team. In *Proc. of the 11th edition of the Ibero-American Conference on Artificial Intelligence, IBERAMIA 2008*, Lisbon, Portugal, October 2008.

[9] J. Silva, N. Lau, J. Rodrigues, J. L. Azevedo, and A. J. R. Neves. Sensor and information fusion applied to a robotic soccer team. In *Proc. of the RoboCup 2009*, Graz, Austria, 2009.

[10] N. Lau, L. S. Lopes, and G. Corrente. CAMBADA: information sharing and team coordination. In *Proc. of the 8th Conference on Autonomous Robot Systems and Competitions, Portuguese Robotics Open - ROBOTICA'2008*, pages 27–32, Aveiro, Portugal, April 2008.

[11] L. Almeida, F. Santos, T. Facchinetti, P. Pedreira, V. Silva, and L. S. Lopes. Coordinating distributed autonomous agents with a real-time database: The CAMBADA project. In *Proc. of the 19th International Symposium on Computer and Information Sciences, ISCIS 2004*, volume 3280 of *Lecture Notes in Computer Science*, pages 878–886. Springer, 2004.

[12] F. Santos, G. Corrente, L. Almeida, N. Lau, and L. S. Lopes. Self-configuration of an adaptive TDMA wireless communication protocol for teams of mobile robots. In *Proc. of the 13th Portuguese Conference on Artificial Intelligence, EPIA 2007*, Guimares, Portugal, December 2007.

[13] F. Santos, L. Almeida, L. S. Lopes, J. L. Azevedo, and M. B. Cunha. Communicating among robots in the robocup middle-size league. In *Proc. of the RoboCup 2009*, Graz, Austria, 2009.

[14] P. Pedreiras, F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, and F. Santos. Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques. In *Proc. of RoboCup 2006*, volume 4020 of *Lecture Notes in Computer Science*, pages 371–383. Springer, 2006.

[15] P. Pedreiras and L. Almeida. *Task Management for Soft Real-Time Applications Based on General Purpose Operating Systems, Robotic Soccer*. Itech Education and Publishing, Vienna, Austria, 2007.

[16] W. Aangenent, J. Best, B. Bukkems, F. Kanters, K. Meessen, J. Willems, R. Merry, and M. Molengraft. Tech united eindhoven team description 2009. Technical report, Control Systems Technology Group, Eindhoven University of Technology, Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, 2009.

[17] R. Hafner, S. Lange, M. Riedmiller, and S. Welker. Brainstormers tribots team description. Technical report, Neuroinformatics Research Group, Institute of Computer Science and Institute of Cognitive Science, University of Osnabrck, 49069 Osnabruck, Germany, 2009.

[18] O. Zweigle, U. Kappeler, H. Rajaie, K. Haussermann, A. Tamke, A. Koch, B. Eckstein, F. Aichele, and P. Levi. 1. rfc stuttgart team description 2009. Technical report, IPVS, University of Stuttgart, 70569 Stuttgart, Germany, 2009.

[19] Y. Kitazumi, S. Ishida, Y. Ogawa, K. Yamada, Y Sato, M. Oki, H. Thoriyama, N. Shinpuku, Y. Takemura, A. Nassiraei, I. Godler, K. Ishii, and H. Miyamoto. Hibikino-musashi team description paper. Technical report, Kyushu Institute of Technology, The University of Kitakyushu, Japan, 2009.

[20] T. Amma, P. Baer, K. Baumgart, P. Burghardt, K. Geihs, J. Henze, S. Opfer, S. Niemczyk, R. Reichle, D. Saur, A. Scharf, J. Schreiber, M. Segatz, S. Seute, H. Skubch, S. Triller, M. Wagner, and A. Witsch. Carpe noctem 2009. Technical report, Distributed Systems Group, University of Kassel, Germany, D-34121 Kassel, Germany, 2009.

[21] H. Zhang, X. Wang, H. Lu, S. Yang, S. Lu, J. Xiao, F. Sun, D. Hai, and Z. Zheng. Nubot team description paper 2009. Technical report, College of Mechatronics and Automation, National University of Defense Technology, China, 410073, 2009.

[22] M. Gholipour, S. Ebrahimijam, H. Rasam Fard, M. Montarezi, A. Mohseni, S. Moein, A. Zaeri, H. Hosseini, M. Yekkefallah, S. Sajjadi, and B. Eskandariun. Mrl middle size team: 2009 team description paper. Technical report, Mechatronics Research Laboratory, Islamic Azad University of Qazvin, Qazvin, IRAN, 2009.

[23] S. Mitri, S. Frintrop, K. Pervolz, H. Surmann, and A. Nuchter. Robust object detection at regions of interest with an application in ball recognition. In *Proc. of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005*, pages 125–130, Barcelona, Spain, April 2005.

[24] R. Hanek, T. Schmitt, and S. Buck. Fast image-based object localization in natural scenes. In *Proc. of the 2002 IEEE/RSJ Int. Conference on Intelligent Robotics and Systems*, pages 116–122, Lausanne, Switzerland, October 2002.

[25] A. Treptow and A. Zell. Real-time object tracking for soccer-robots without color information. *Robotics and Autonomous Systems*, 48(1):41–48, August 2004.

[26] S. Mitri, K. Pervolz, H. Surmann, and A. Nuchter. Fast color independent ball detection for mobile robots. In *Proc. of the 2004 IEEE Int. Conference on Mechatronics and Robotics*, pages 900–905, Aechen, Germany, September 2004.

[27] G. Coath and P. Musumeci. Adaptive arc fitting for ball detection in RoboCup. In *Proc. of the APRS Workshop on Digital Image Computing, WDIC 2003*, pages 63–68, Brisbane, Australia, February 2003 2003.

[28] H. Lu, H. Zhang, and Z. Zheng. Arbitrary ball recognition based on omni-directional vision for soccer robots. In *Proc. of RoboCup 2008*, 2008.

[29] M. Nixon and A. Aguado. *Feature Extraction and Image Processing*. Reed Educational and Professional Publishing Ltd, Linacre House, Jordan Hill, Oxford OX2 8DP 225 Wildwood Avenue, Woburn, MA 01801-2041, first edition, 2002.

[30] G. Bradski and A. Kaehler. *Learning OpenCv, Computer Vision with the OpenCv Library*. OReilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, first edition, September 2008.

[31] I. Pinheiro. Automatic calibration of the cambada team vision system. Master's thesis, Universidade de Aveiro, 2008.